Система за оценка на знанията на студентите по програмиране

Николай Киров
(Работна версия)

26 септември 2012 г.

1 Увод

Преподаването на програмиране на студентите от началните курсове е една трудна задача за всеки преподавател. Това се обуславя от спецификата на програмирането. ...

Има се предвид 3 основни курса, които се четат обикновено от първи до трети семестър – *Въведение в програмирането*, *Обектно-ориентирано програмиране* и *Структури от данни*. Тези имена са примерни,...

Усвояването на учебния материал е свързано с придобиване на практически умения за писане на програми. Затова навсякъде преподаването на програмиране е съставено от две части – лекции и упражнения. Лекциите се провеждат в лекционна зала, а лабораторни упражненията – в компютърна зала. В някои университети се добавя и трети елемент – т.н. теоретични упражнения в лекционна зала. Хорариумите варират от $30+30$ до $60+30+30$ учебни часа за семестър.

2 Точкова система за оценка

В началото на всеки семестър на студентите се обявяват правилата за оценка на придобитите знания и умения по конкретната учебна дисциплина. Тези правила са формулирани ясно и публикувани в Интернет на сайта на курса [5].

Следната таблица дава съответствието на придобитите точки и получената оценка по шестобалната система. Вдясно е дадена таблица от курс по програмирене на университет в САЩ [12]:

Точки	оценка
0 - 49	2
50 - 59	3
60 - 75	4
76 - 89	5
90 - 100	6

Total	Grade
>=93	A
90 to <93	A–
87 to <90	B+
83 to <87	B
80 to <83	B–
75 to <80	C+
70 to <75	C
65 to <70	C–
63 to <65	D+
60 to <63	D
50 to <60	D–
0 to <50	F

2.1 Текуща оценка

Университетската система в Англия и САЩ дава възможност на студента да получи (крайна) оценка по дадена учебна дисциплина в края на семестъра, без да се явява на т.н. семестриален изпит. За тази цел по време на семестъра се оценява работата на студента и при набиране на определен брой точки (или кредити), студентът получава т.н. текуща оценка, която в края на семестъра става крайна оцинка. При някои други системи, когато семестриалният изпит е задължителен, той дава само определена част от крайната оценка, а другата част е текущата оценка.

Университетската система в Англия и САЩ дава възможност на студента да получи (крайна) оценка по дадена учебна дисциплина в края на семестъра, без да се явява на т.н. семестриален изпит. За тази цел по време на семестъра студентът набира точки (от изпълнение на различни задачи), и в края на семестъра при събиране на определен брой точки, студентът получава т.н. крайна текуща а, която става крайна оценка по учебната дисциплина. При някои други системи, когато семестриалният изпит е задължителен, той дава само определена част от крайната оценка, а другата част е текущата оценка.

Текущата оценка се формира от:

- Три теста ($10 + 10 + 20 = 40$ точки)
- Три контролни ($10 + 10 + 10 = 30$ точки)
- Три домашни ($10 + 10 + 10 = 30$ точки)

В някои университети се изисква да се постави отделна оценка от упражненията. Тогава може да се използва следната таблица:

- Три контролни ($10 + 10 + 50 = 70$ точки)
- Три домашни ($10 + 10 + 10 = 30$ точки)

Обикновено в края на семестъра допълнително се дават т.н. бонус точки по субективна преценка на преподавателя. Те не са повече от 10 и могат да бъдат:

- за редовно присъствие на занятия;
- за участие и добро представяне в състезания по програмиране;

- за отговори на въпроси, задавани по време на лекциите.

2.2 Семестриален изпит

В два случая студентът трябва да се яви на семестриален изпит:

- ако е събрал под 50 точки на текущ контрол или
- ако иска да повиши текущата си оценка.

Семестриалният изпит се състои от:

- Трети тест (40 точки)
- Трето контролно (20 точки)
- Писмен изпит (20 точки)
- Събеседване (20 точки)

Същото важи и ако се налага да се постави оценка от упражненията, като е в сила следната таблица:

- Три домашни ($10 + 10 + 10 = 30$ точки)
- Трето контролно (70 точки)

Обикновено последното учебно лекционно занятие на семестъра е тест (Трети тест), а упражненията са контролна работа (Трето контролно). Резултатите от тях се зачитат за получаване на текуща оценка, а също така и при семестриалния изпит, ако студентът се яви на такъв изпит. На датата на семестриалния изпит студентът може (по желание) да прави тест и да се яви на контролно. Ако получи повече точки на това явяване, се зачитат тези точки, в противен случай се вземат точките от Трети тест и/или Трето контролно от предишното явяване. На следващите поправителни изпити важат същите правила.

3 Тестове

Тестовете са от типа "множествен избор" (multiple choice), всеки въпрос има предложения за 4 отговора, маркирани с а), б), в) и г). За всеки въпрос са възможни всички вариации с повторения на елементите вярно-невярно (в-н), т.е.

- Всички са верни `вввв`,
- Един е верен `вввн, ввнв, внвв, нввв`,
- Два са верни `ввнн, внвн, виив, нввн, нвнв, ннвв`,
- Три са верни `внин, нвин, ннвн, ннив`,
- Няма верен `нннн`.

Броят на вариантите е $4^2 = 16$. За всеки отговор студентът има 3 възможности за определянето му:

- знам, че това е (верен) отговор на въпроса;

- знам, че това е неверен отговор на въпроса, или по-точно, предложението не е отговор на въпроса;
- не мога да преценя, не знам дали предложението е отговор на въпроса.

Всеки студент получава лист хартия A4 с теста. Всички тестове са различни, което до голяма степен елиминира елемента на преписване при попълване на теста.

3.1 Подготовка

Подготовката на тестовете включва формулиране на въпросите и подбор на отговорите – верни и неверни. По-точно казано, предложения за отговори на въпросите, като някои от тях са (верни) отговори, а други не са. Тъй като всеки индивидуален тест съдържа по 4 предложения, то всеки въпрос трябва да има най-малко 4 отговора. За да се избегне преписване от съседа, е добре да се подготвят поне 12 предложения за отговори на всеки въпрос. Въпросите и отговорите формират текстов файл със строго определена структура (Приложение 7.1).

Този файл служи за вход на програма, която генерира индивидуалните тестове – толкова, колкото е броят на студентите (Приложение 7.1). Използва се генератор на случайни числа за избор и на въпросите и на отговорите на всеки индивидуален тест. Всеки такъв тест има уникален (обикновено) 4-цифров номер. Най-често един индивидуален тест съдържа 10 или 12 въпроса, което се определя главно от размера на въпросите и отговорите, като един тест се събира точно на една страница. Като опция най-отдолу на страницата може да се се даде общия брой на верните отговори. Програмата генерира LaTeX [7] файл със зададен брой тестове, след което със системата MiKTeX [8] се произвежда `ps` или `pdf` файл, готов за отпечатване (Приложение 7.1). Програмата генерира и файл с отговорите на всеки индивидуалин тест (Приложение 7.1).

Поне една седмица преди датата на теста въпросите и по два примерни отговора (верен и грешен) се публикува в Интернет на сайта на курса (Приложение 7.1). Така студентите имат възможност да се запознаят с теста предварително и при нужда да се свържат с преподавателя за изясняване или консултация.

Последният, трети тест съдържа и въпроси от предишните два теста, често леко променени – сменени имена на променливи, построяване на отрицания на някои от твърденията и др. подобни.

3.2 Провеждане

Преди да се раздадат индивидуалните тестове се припомнят правилата за провеждане на теста:

- По време на теста студентите могат да използват лекции, учебници и всякакви други печатни материали
- На отделен лист хартия или използвайки даден шаблон за отговори, студентите трябва срещу всяко предложение за отговор поставят или положителен отговор (да, вярно, истина; yes, correct, true) или отрицателен отговор (не, невярно, неистина; no, incorrect, false) или неутрално (–, не знам; или нищо; I don't know).

- Ако има питания за неясноти по теста, преподавателят отговаря на въпросите персонално.

Когато един и същи въпрос е зададен от няколко души, той се обявява на всички, заедно с отговора. Понякога, използвайки мултимедиината проектор, показвам слайдовете от съответната лекция, където явно или по-често неявно се съдържа отговорът на зададения въпрос.

По-подробно за феномена преписване. Класическият подход е когато студентът прави контролно, тест, явява се на изпит и трябва да покаже какво е научил, му се забранява да ползва външни източници на информация, т.е. учебният материал се запомня и после или се възпроизвежда или трябва да се приложи – например за решаване на задачи или писане на програми. В специфичната дейност програмиране, която е основна цел като придобито умение по тези дисциплини, крайният продукт е компютърна програма. Написването и е до голяма степен творчески процес и ограничаването на използваната информация е напълно безсмислено. Например работодател от ИТ индустрията да даде задача на програмиста за написване на софтуер и да не му дава да използва литература – напротив, изискването обикновено е да се намери литература и да се създаде софтуера според най-новите тенденции в специфичната област. Смятам, че този подход трябва да се прилага още на "студентската скамейка".

Целта от провеждане на теста е не само да проверим знанията на студентите, а и да ги накараме да отворят учебника, да потърсят конкретна информация там, да направят логическата връзка между въпросите на теста и написаното в учебника.

Каква радост за окото на преподавателя, когато всички студенти в залата са отворили учебници и четат ли четат!

Времето да провеждане на теста е 2 учебни часа или един учебен блок (една лекция). Когато студентът е готов, той предава двата листа хартия – условията на индивидуалния тест и неговите отговори (Приложение 7.1).

3.3 Проверка от преподавателя

При проверката на индивидуален тест всеки въпрос 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 отговор;
- 0 при отговор "не знам".

За целия индивидуален тест се прави сума на броя на точките по всеки въпрос. Нека тестът съдържа N въпроса, и нека един конкретен тест да е събрал t тестови точки ($t \leq 4N$). Ако правенето на теста носи максимално M изпитни точки, то този тест дава $e = \dfrac{tM}{4N}$ изпитни точки на студента. Ако e не е цяло, то се взема най-малкото цяло число, по-голямо от e – закръгляване нагоре, т.е. в полза на студента (Приложение 7.1).

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."[6]. Аргументът ми за неспазване на това правило е, че програмирането

е специфична дейност и знанията в тази област трябва да са категорични – програмистът трябва да си дава ясна сметка каво 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 ила загуба на космически кораб.

3.4 Проверка от студента

В началото на следващата лекция се обявяват резултатите от теста и се връщат тестовете на студентите. Всеки студент 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 от студентите е те 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 някои въпроси, също така може да има не съвсем ясни предложения за отговори, практиката ми е да приемам мненията на студентите, които интерпретират по по-различен начин някой въпрос или някой отговор и да увеличавам изпитните точки. След проверката понякога събирам всички тестове (условия и отговори), друг път ги оставям на студентите. Практиката ми е да не връщам тестове на студенти, които отсъстват от лекцията с проверката.

4 Домашни работи

За един семестър има предвидени 3 домашни работи. Студентите имат между 1 и 2 седмици от задаване на домашното до представянето му за проверка.

4.1 Подготовка

Подбирам определен брой задачи, в зависимост от броя на студентите и същността на учебния материал. Някои домашни са пряко свързани с упражненията от съответния учебник, други са разширения на примери, обсъждани на лекции, има и такива, за които студентът трябва да намери и разучи материал от Интернет (напр. алгоритъм).

Определянато на кой номер задача от списъка ще решава конкретен студент зависи от неговия факултетен номер. Ако има n подготвени домашни, а факултетния номер на студента е F, то номерът на задачата му се получава като остатък от целочислено деление – аритметичната операция в C и C++: `f%n` или в математиката модул: $f \bmod n$. По този начин обикновено една задача се пада на повече от един студент. Това стимулира съвместна работа в екип за решаване на задачата, което е важен елемент от подготовката на програмистите. Разбира се, възможността един студент да направи домашното, а другите да го препишат остава.

Списъкът със задачите за домашна работа се публикува в Интернет на сайта на курса.

4.2 Проверка

Готовите програми, като текст на програмата (source code) се изпращат на ръководителя на упражненията по e-mail. Те се проверяват и оценяват без присъствието на сту-

дентите. Понякога, а за някои курсове като правило, водещият упражненията изисква студентът сам да представи домашната си работа, като донесе текста на програмата, компилира го, тества програмата с примери, зададени от ръководителя и обясни алгоритмите и реализациите им. Така оценката може да се намали, ако е видно, че студентът е преписал програмата и не си е изяснил работата и.

5 Контролни работи

Контролните работи се правят в часовете за упражнения и представляват задание за написване на програма на компютър, компилиране и тестване на написаната програма с примери на преподавателя. Също така студентът трябва да може да обясни алгоритмите и техниките, които е използвал при написване на програмата.

Това е може би най-важният елемент от оценяването, защото тук се концентрират и знанията от лекциите и опита от самостоятелните (домашни) работи.

6 Писмен изпит и събеседване

Писменият изпит е последна фаза от семестриалния изпит. Студентът вече е събрал точки от теста и контролното – максимум 60, което е достатъчно за оценка добър (ако се откаже от писмения изпит и събеседването) и минимум 10, което е достатъчно за обща оценка среден, ако на писмения изпит и събеседването събере максимален брой точки (40).

Студентът си тегли въпрос по класическата технология. За тази цел може предварително да е даден конспект с въпроси или, както напоследък практикувам, един въпрос представлява учебния материал от една лекция или от половин лекция. Възможни са и варианти – един въпрос на принципа "избери си въпросът, който знаеш най-добре" и един въпрос, избран от преподавателя. Така се елиминира случая "един въпрос не бях учил и той ми се падна".

Събеседването се състои в обсъждане на написаното на хартията от студента. Обикновено подчертавам написани твърдения, които са неясни, двусмислени, не съвсем верни, верни с допълнение или неверни, като се опитвам да накарам студента да мисли и да изясни какво е имал предвид, когато е писал тези твърдения. Ако има непълнота в текста, мога да поискам да допише текста по конкретен въпрос, да даде пример или да напише малък програмен код.

Обикновено събеседването започва с аритметика – колко точки студентът е събрал дотук и за каква оценка може да се бори. Например, ако има събрани 45 изпитни точки, то 5 точки от тази фаза (което означава четвърт въпрос от двата) студентът получава среден, ако успее да направи 15 точки (един въпрос с малки пропуски), може да получи добър. Максимална оценка мн.добър ще се достигне (35 точки) с перфектно написани въпроси и евентуални малки грешки при събеседването. На тази сесия, студентът не може да се бори за отлична оценка поради слабите си постижения от теста и контролното.

7 Практическо прилагане на системата за оценка

Идеята за подобна система за оценка възникна през 1999 година, когато започнах да преподавам програмиране в ЮЗУ Неофит Рилски, Благоевград. През годините съм променял много детайли, и в този вид системата се прилага от 4 години в НБУ за 3 учебни дисциплини [5]:

- Въведение в програмирането (C++)
- Обектно-ориентирано програмиране
- Структури от данни

Варианти на системата са прилагани в БСУ по учебни дисциплини Програмиране и използване на компютрите I и II за инженерни специалности, а също така и в ЮЗУ по Операционни системи за специалности математика и информатика.

Важен елемент при използване на тестове е анализ на резултатите от конкретен тест, където могат да се направят статистически обработки, да се оценят въпросите и отговорите в теста и др. За съжаление ...

7.1 Публикуване в Интернет на резултатите на студентите

Въпросът за публикуване на резултатите на студентите е спорен. От една страна когато колегите могат да видят резултатите

Моя опит показва, че публичното показване на резултатите на студентите е ... (Приложение 7.1)

Литература

[1] Н. Киров, Преподаването по програмиране във висшите училища, Математика и математическо образование, 2001.

[2] Н. Киров, Сборник от учебни материали по Програмиране и структури от данни, Деметра, София, 2004.

[3] Н. Киров, Сборник от учебни материали по Въведение в програмирането, Деметра, София, 2003.

[4] Бижков, Г. Теория и методика на дидактическите тестове, "Просвета", С. 1992 г.

[5] Н. Киров, Учебни материали в Интернет.
`http://www.math.bas.bg/~nkirov/teach_bg.html`

[6] Наръчник по оценяване, Център за оценяване, НБУ, 2007.

[7] LaTeX -– A document preparation system `http://www.latex-project.org/`

[8] MiKTeX project page `http://miktex.org/`

[9] Cay Horstmann, Computing Concepts with C++ Essentials, Third Edition, John Wiley & Sons, 2003.

[10] Michael Goodrich, Roberto Tamassia, David M. Mount, Data Structures and Algorithms in C++, Wiley, 2004.

[11] H.M.Deitel, P.J.Deitel, C++ How to Program, Fourth edition, Prentice Hall, 2002.

[12] Sillabus – Data Structure and Algorithm, Department of Computer Science, California State University San Marcos
`http://courses.csusm.edu/cs311ah/Pages/Syllabus.htm`

Приложения

Приложение 1. Подготвени от преподавателя въпроси и отговори

```
\eng{3}
Mark with ‘‘yes’’ the statements in which the loop body
executes exactly once. The integer variable \verb|beep| has value \verb|1|.
- \verb|while(beep > 0) beep++;|
- \verb|while(beep < 0) beep++;|
- \verb|while(beep >= 0) beep++;|
+ \verb|while(beep == 1) beep++;|
+ \verb|while(beep > 0) beep--;|
- \verb|while(false) beep--;|
+ \verb|do beep++; while(false);|
+ \verb|do beep++; while(beep < 0);|
+ \verb|do beep++; while(beep == 1);|
+ \verb|do beep++; while(beep == 0);|
- \verb|do beep++; while(beep == 2);|
- \verb|do beep--; while(true);|
+ \verb|do beep--; while(beep > 0);|
- \verb|do beep--; while(beep == 0);|
- \verb|for(beep = 1; beep < 0; beep++) cout << beep;|
+ \verb|for( ; beep == 1; beep++) cout << beep;|
- \verb|for( ; beep >= 0; beep--) cout << beep;|
+ \verb|for( ; beep > 0; beep--) cout << beep;|

\eng{4}
We have the following function declaration:
 \vspace{-2mm}\begin{verbati }
int verb(double&, double);
 \end{verbati }\vspace{-5mm}
Mark correct/incorrect statements. We suppose that a double
variable \verb|mode| is defined and initialized with \verb|0.0|.
- \verb|cout << verb(10, mode);|
- \verb|if(verb(2, mode) < 0) mode = 2;|
- \verb|if(2*verb(-1, mode) > mode) mode = 1;|
- \verb|cout << verb(0, mode + 1);|
+ \verb|mode = verb(mode, mode*mode);|
+ \verb|cout << verb(mode, 2.5);|
+ \verb|cout << verb(mode, mode - 2);|
+ \verb|double x = verb(mode, mode);|
+ \verb|mode = 2*verb(mode, -mode);|
- \verb|int k = verb(-mode, 0);|
- \verb|if (verb(2*mode, mode) != 2) mode++;|
- \verb|cout << verb(0, 2.5);|
```

Приложение 2. Текст на част от LaTeX файла с индивидуалните тестове, генериран от програмата

```
\newpage\thispagestyle{empty}\par \hrule \begin{center}
 {\bf\large \fbox{2007/2008}}\hfill
{\bf\large\verb|Final Test|}
\hfill{\bf\large \fbox{ NBU }}\\
{\bf\large NETB101 Introduction to Programming (C++)}\\[4pt]
{\bf \large \fbox{T E S T \ \ No. 1374
}}\\
\end{center}\par\vspace{-4mm}
\begin{multicols}{2}
%20  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\par\smallskip\hrule\par\smallskip

\medskip  %%%  7 %%%
{\bf 1.}\ We have the following variable definitions:
 \vspace{-2mm}\begin{verbati }
string num = "2008";
const string ABC = "abc";
 \end{verbati }\vspace{-5mm}
Mark the syntax correct/incorrect statements.

{\bf a)}\ \verb|cout << num.substr(0,2);|

{\bf b)}\ \verb|string s = ABC.length();|

{\bf c)}\ \verb|num=ABC + ABC.substr(0,1);|

{\bf d)}\ \verb|cout << length(ABC);|
\par\smallskip\hrule\par\smallskip

\medskip  %%%  4 %%%
{\bf 2.}\ We have the following function declaration:
 \vspace{-2mm}\begin{verbati }
int verb(double&, double);
 \end{verbati }\vspace{-5mm}
Mark correct/incorrect statements. We suppose that a double
variable \verb|mode| is defined and initialized with \verb|0.0|.

{\bf a)}\ \verb|cout << verb(mode, mode - 2);|

{\bf b)}\ \verb|cout << verb(mode, 2.5);|

{\bf c)}\ \verb|cout << verb(0, 2.5);|
```

Приложение 3. Всички въпроси и по 2 отговора на всеки въпрос от сайта на курса

```
The questions of Final Test
with two model answers - one correct (yes) and one incorrect (no)

Mark the syntax correct/incorrect loop statements.
Variables t, y, h and k have int type.
(yes) for(t = -1; t < 4; t++) cout << t;
(no) for{y = 1; y < 10; y++} cout << h;

We have the following function declaration:
    int verb(double&, double);
Mark the correct/incorrect statements. We suppose that a double variable
mode is defined and initialized with 0.0.
(yes) cout << verb(mode, 2.5);
(no) cout << verb(0, 2.5);

Mark with "yes" infinite loops and with "no" finite loops.
The value of the integer variable far is obtained from the input stream.
(yes) do far++; while (far != 0);
(no) do far--; while(far > -3200);

Mark with "yes" the statements in which the loop body executes exactly
once. The integer variable beep has value 1.
(yes) while(beep == 1) beep++;
(no) while(false) beep--;

Mark the correct/incorrect statements about the class Employee
(as it is defined in our textbook) and an object merry of this class.
(yes) merry.set_salary(2000);
(no) Employee maya = set_salary(2000.00);

We have a class
  class Prod {
  public:
    Prod();
    Prod(string n, int s);
    void read();
    string get_name() const;
    void print() const;
  private:
    string name;
    int sc;
  };
```

Приложение 4. Индивидуален тест

2007/2008 Final Test **NBU**

NETB101 Introduction to Programming (C++)

T E S T No. 1374

1. We have the following variable definitions:

```
string num = "2008";
const string ABC = "abc";
```

Mark the syntax correct/incorrect statements.

a) `cout << num.substr(0,2);`
b) `string s = ABC.length();`
c) `num=ABC + ABC.substr(0,1);`
d) `cout << length(ABC);`

2. We have the following function declaration:

```
int verb(double&, double);
```

Mark correct/incorrect statements. We suppose that a double variable `mode` is defined and initialized with `0.0`.

a) `cout << verb(mode, mode - 2);`
b) `cout << verb(mode, 2.5);`
c) `cout << verb(0, 2.5);`
d) `if(2*verb(-1, mode) > mode) mode = 1;`

3. We have the following variable definitions:

```
int ten = 10;
int nine = 9;
int eight = 8;
```

Calculate the arithmetic expression and mark it "yes" if it has value 1.

a) `ten % nine / 2`
b) `nine + eight / nine`
c) `ten + eight / nine`
d) `ten / nine % 2`

4. We have:

```
double a, x;
bool flag1, flag2;
```

Does C++ run-time system compute the following logical expressions using lazy evaluations?

a) `flag2 && (a <= x)`
b) `!(x >= 0)`
c) `cin.fail() || (a < 0)`
d) `(x >= 0) && flag1`

5. Mark with "yes" the equivalent logical expressions (1) and (2).

a) (1) `country == "USA"` (2) `! (country != "USA")`
b) (1) `country == "USA" && state != "AK"`
(2) `! (country != "USA" || state == "AK")`
c) (1) `state == "AK" && state == "HI"`
(2) `! (state == "HI")`
d) (1) `country == "USA" && state != "HI"`
(2) `! (country == "USA") && state == "HI"`

6. Mark with "yes" infinite loops and with "no" finite loops. The value of the integer variable `far` is obtained from the input stream.

a) `do far--; while((far != 0) or (far == 0));`
b) `do far++; while ((far != 0) or (far == 0));`
c) `while(false) far++;`
d) `while(far == 32000) far++;`

7. Mark the syntax correct/incorrect loop statements. Variables `t`, `y`, `h` and `k` have `int` type.

a) `for( k++ ) cout << k;`
b) `for{h = 1; h > 0; h--} (cout << h);`
c) `for(h == 1;) return;`
d) `for{y = 1; y < 10; y++} cout << h;`

8. Mark with "yes" the statements in which the loop body executes exactly once. The integer variable `beep` has value `1`.

a) `do beep--; while(beep == 0);`
b) `do beep++; while(false);`
c) `while(beep >= 0) beep++;`
d) `while(beep > 0) beep++;`

9. Mark the correct/incorrect statements about the class `Employee` (as it is defined in our textbook) and about an object `merry` of this class.

a) `string n = merry.get_name();`
b) `merry = Employee("Mary", 500.0);`
c) `get_salary(3000.0);`
d) `Employee Zoe(1000.0);`

10. We have a class

```
class Prod {
public:
   Prod();
   Prod(string n, int s);
   void read();
   string get_name() const;
   void print() const;
private:
   string name;
   int sc;
};
```

and a global object `p` of this class. Mark the correct/incorrect statements in the body of `main` function.

a) `cin >> p.sc;`
b) `cout << p.print();`
c) `Prod q;`
d) `string str = p.get_name();`

11. Mark the correct/incorrect assertions about vectors.

a) The size of a vector is set in the definition and cannot be changed.
b) The member-function `pop_back()` removes the last element of a vector.
c) The statement `vector<int> b[2];` defines a vector of two elements.
d) A vector is a collection of data items of the same type.

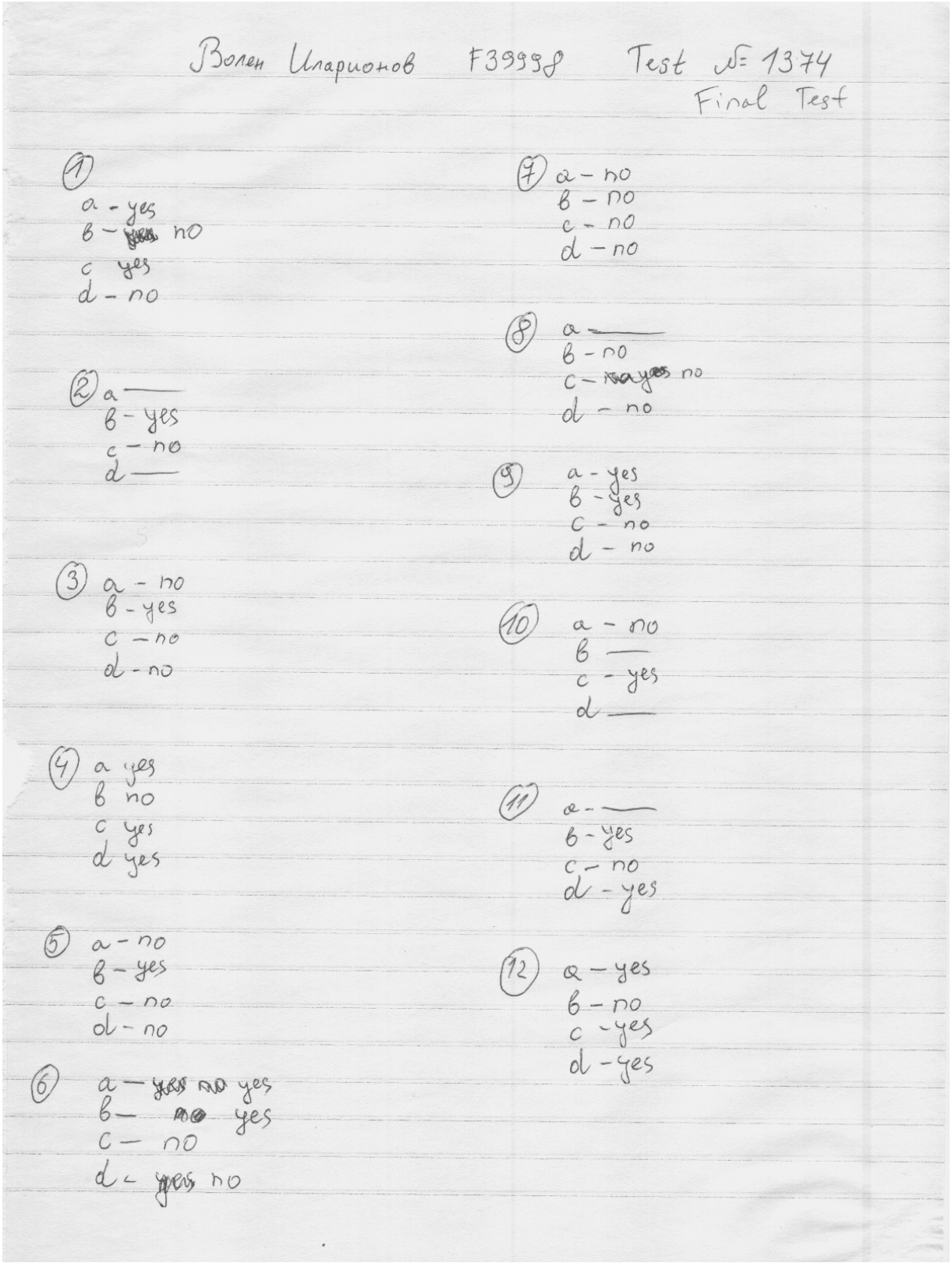
12. We have a vector and values of its elements:

```
vector<int> sol(2);
sol[0] = 1;
sol[1] = 3;
```

Mark correct/incorrect statements.

a) `cout << (sol[0] + sol[1]) / 2;`
b) `cout << sol.pop_back();`
c) `double x = 4 * sol[1];`
d) `cout << sol[sol[1] - 2];`

Приложение 5. Решение на тест, дадено от студент



Волен Иларионов F39998 Test №= 1374
Final Test

1. a - yes
   b - ~~yes~~ no
   c yes
   d - no

2. a ——
   b - yes
   c - no
   d ——

3. a - no
   b - yes
   c - no
   d - no

4. a yes
   b no
   c yes
   d yes

5. a - no
   b - yes
   c - no
   d - no

6. a — ~~yes no~~ yes
   b — ~~no~~ yes
   c — no
   d — ~~yes~~ no

7. a - no
   b - no
   c - no
   d - no

8. a ——
   b - no
   c - ~~no yes~~ no
   d - no

9. a - yes
   b - yes
   c - no
   d - no

10. a - no
    b ——
    c - yes
    d ——

11. a - ——
    b - yes
    c - no
    d - yes

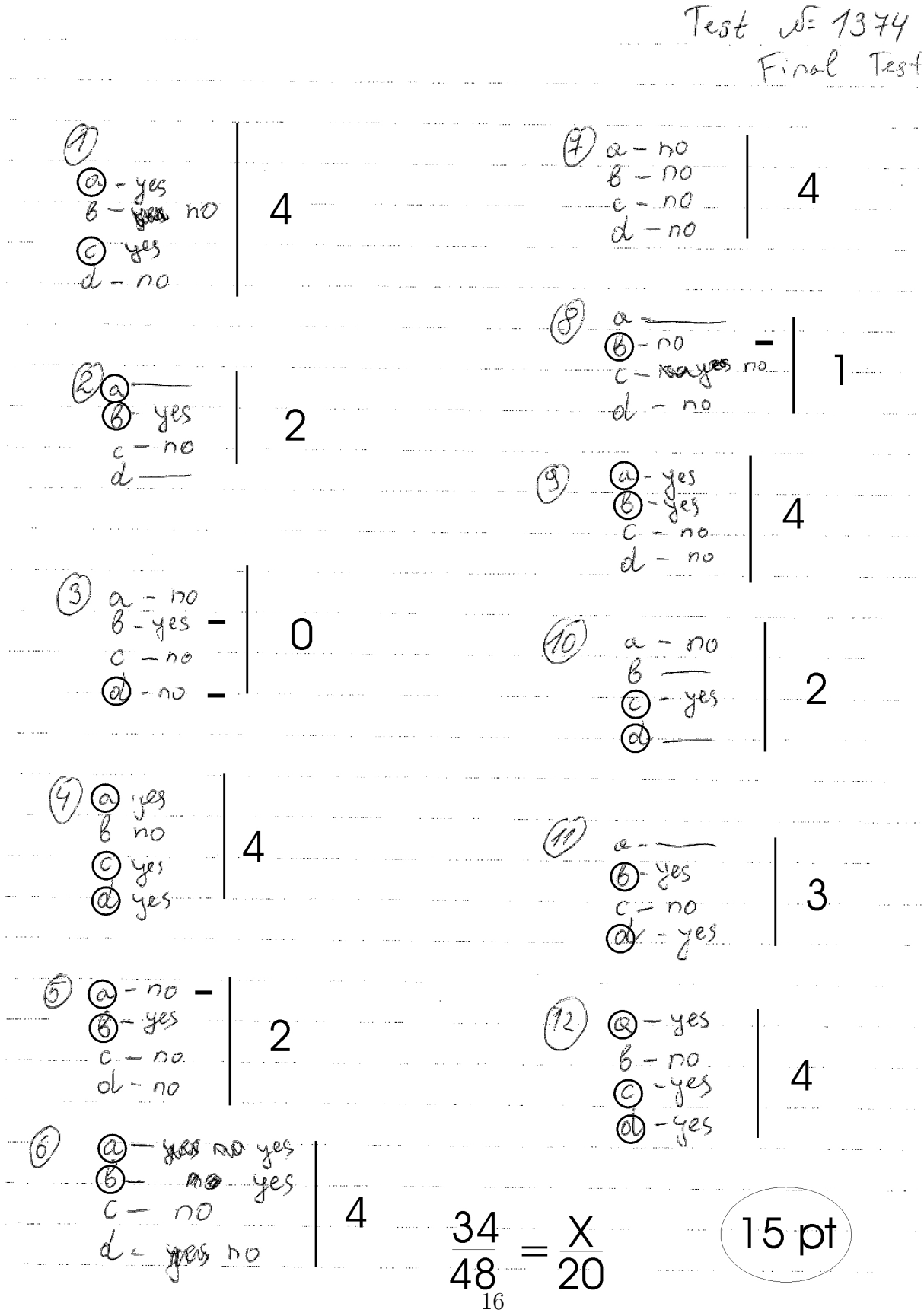
12. a - yes
    b - no
    c - yes
    d - yes

Приложение 6. Генерирани от програмата решения на тестовете

Test 3 – NETB101, 1.2008

Test No.1374	1.ac	2.ab	3.d	4.acd	5.ab	6.ab	7.	8.b	9.ab	10.cd	11.bd	12.acd	(22)
Test No.1409	1.bd	2.bc	3.	4.abd	5.abc	6.ab	7.d	8.ab	9.b	10.bcd	11.d	12.	(20)
Test No.1486	1.abc	2.b	3.abcd	4.ac	5.d	6.ac	7.	8.d	9.d	10.c	11.c	12.ad	(19)

Приложение 7. Попълнен от студента и проверен от преподавателя



Приложение 8. Резултати на студенти

Това са действителни резултатите на студенти от 1 курс (2007), NETB101, програма Мрежови технологии, Трети тест за максимум 20 изпитни точки. 38 студенти са се явили на теста, 15 са получили 10 и повече точки, което съответства на оценка 3 и повече. 23 са слабите оценки, като 10 от тях (с точки 7, 8 и 9) са близо до тройката.

точки	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
брой студенти	0	1	0	3	4	3	3	2	5	3	1	0	3	0	1	4	0	4	0	2	0
оценка	2	2	2	2	2	2	2	2	2	2	3	3.5	3.5	4	4	4.5	5	5.5	5.5	6	6