СТИЛ НА ПРОГРАМИРАНЕ В ОБУЧЕНИЕТО[1]

Теодоси Теодосиев

Шуменски университет “Епископ К.Преславски”
Факултет по математика и информатика
ул. “Университетска” 115, Шумен
t.teodosiev@fmi.shu-bg.net

Резюме: *Предлаганата работа разглежда необходимостта от обучение в стил на програмиране на новаците. Коментира се какво представлява стила на програмиране. Представена е ползата от добрия стил. Обоснована е нуждата обучението в стил да започне паралелно с изучаването на езика за програмиране.*

Ключови думи: *програмиране, обучение, стил на програмиране, език за програмиране.*

1. Увод. Достатъчно ли е да програмираме правилно и ефективно?! Работейки над програмата, програмистът, особено начинаещият, е длъжен да се съобразява с факта, че програмата е предназначена от една страна за потребителя, а от друга – за самия програмист. Текстът на програмата е нужен преди всичко на самия програмист, а също и на други хора, с които той съвместно работи над проекта. За да бъде работата ефективна, програмата трябва да е лесно четивна, нейната структура трябва да съответства на структурата и алгоритъма на решаваната задача.

2. Стил на програмиране. Не съществува съвкупност от правила за написване на програма следването, на които да създава качествена и несъдържаща грешки програма. “Стил на програмиране трябва да се разбира като тълкуване на личността на набор от правила и прилагането им при писане на изходния код, за да се постигне дадена цел “ [10], така че изходният код да е четивен и разбираем. Програмната четивност е с положителен корелационен коефициент при статистическата обработка на програмите на участниците в IOI’94, направена от Григас [8] въпреки, че не се оценява и времевите ограничения по време на олимпиадата. Това потвърждава идеята, че програмната четивност има положително отражение върху нейната коректност. Същото се наблюдава и при други параметри, с положителен ефект върху четивността (например, коментари).

Може да се каже, че под концепцията за стил на програмиране се разбира всичко, свързано с програмната яснота, простота и всеобщност.

[1] Работата е финансирана по проект № РД-05-151/25.02.2011 от фонд “Научни изследвания” на ШУ

Разработена е класификация за стил на програмиране, но все още е трудно да се определят точно елементите на стила на програмиране и за изразяване на съответствието им количествено [3]. Дефинициите на константи е признак на добър стил на програмиране. Коефициентите на корелация, подсказват, че е по-добре да се разчита на прости и добре познати елементи на езика за програмиране (като цикъл for), а не да се използват допълнителни, но по-сложни или по-малко общи елементи, особено ако те не са необходими за решаването на проблема.

Широко прието е, че стила на програмиране има известно влияние върху програмната коректност и ефективността на работата на програмиста.

Едно от основните изисквания е последователност навсякъде в програмата: в форматиране на текста, в наименованието на обекти, в обработката на данни, в структурите за контрол и т.н.

Другите основни изисквания са: чисто и ясно форматиране на текст, отстъп за разкриване на програмната структура, правилното използване на коментари, описателни имена, основателно избиране и използване на структури от данни, структурна програма (отделни групи от изчислителни стъпки са разделени в отделни функции).

Строг критерий за оценка на степента на съответствие на програмата на добрия стил на програмиране не съществува. Заедно с това, достатъчен е един поглед, за да оценим дали програмата съответства на добрия стил на програмиране или не.

Не може да се отрече възможността за конфликт между удобството и ефективността, но това не е закономерно следствие. Струва си да се уточни, до каква степен интересите на човека и машината съвпадат и да се види, какви технологии можем да измислим в полза на всички нас.

Програмисткият стил се оценява като функция на модулността, типизацията, яснотата, независимостта, ефективността и надеждността.

За високата надеждност спомагат добрите програмни спецификации, метода за програмиране (при добрия метод се програмира с по-малко грешки), методите за настройка и тестване (чрез тях допуснатите грешки се откриват по-лесно и бързо). За надеждността спомага също и правилният стил на програмиране (добрата документация, включените средства за самопроверка и т.н).

“Да бъдеш отличен програмист – означава да си способен да разработваш по-ефективни и достоверни програми и да знаеш как да го правиш ефективно. Към това се отнася икономията на клетки от паметта или машинни цикли, а също избягване на сложности, които увеличават разсъжденията, необходими за удържане на строго интелектуалния контрол над разработките. Според мен, за това е нужно усъвършенстване на

математическите навици, при това аз използвам думата „математика” в смисъл „изкуство и наука за ефективни разсъждения”. В действителност задачата за разработка на висококачествени програми и построяване на висококачествени доказателства са толкова сходни, че аз не мога да ги различа: аз не виждам смислени различия между методологията на програмирането и математическата методология в общи линии.” Такава интересна аналогия между програмирането и математиката (в частта за ефективността и стила) прави в [2] един от корифеите в областта на информатиката Е. Дейкстра. Пак според него, “програмистът е длъжен да демонстрира, че неговата програма има исканите свойства. Ако тази мисъл се появи много късно, навярно няма да може да се справи с тази задача. Само, ако тази цел влияе на разработката това може да стане” [3]. Основният акцент се поставя върху правилността и ефективността на програмата, за което са нужни математически навици и предварителна работа преди да седнем пред компютъра.

Следването на правилата на добрия стил на програмиране значително намаляват вероятността от появяване на грешка на етапа на набиране на текста, прави програмата лесно четивна, това на свой ред облекчава процеса на проверка и внасяне на изменения. Много време отнема на програмистите отстраняването на грешки. Намирането и отстраняването на синтактични грешки става много лесно след малко практика, тъй като компилаторът ни дава представа за това как и къде е сбъркано. Когато проверяваме дадена програма преминаваме от ролята на съставител към тази на читател на програмата. Ето кога, добрият стил и коментари могат да ни спестят много трудности.

3. Обучението по програмиране. Трудностите в началното обучение по програмиране са:

✓ Алгоритмични – трудно детайлизиране на алгоритъма. Традиционното обучение е по готови алгоритми и не достатъчно стимулира обучаемите в мислене. Обучаемите трудно извличат информация от текст. Проблемите на четенето с разбиране все по-ясно се виждат и в резултатите от международните изследвания на нашите ученици (TIMSS, PISA).

✓ Инструментални – изборът на език за програмиране. Този въпрос е почти решен, от гледна точка на масовост с изборът на C/C++. Този избор обаче, не е безспорен и на редица места се търси алтернатива.

Както казва в [1] Дейкстра „Днешното кодиране изисква огромно внимание и неизменен талант към точността, то е трудоемко и затова трябва да се отлага до тогава, докато не сте максимално уверен в това, че програмата, към чието кодиране пристъпвате е същата, към която се стремите”.

Един проблем (много често при ръководителите на софтуерни фирми) е погрешното разбиране, че програмистът трябва да произвежда код, а не да решава проблеми, за което използва кода.

Задачата на програмиста не е програмата, съставена от него, а класът възможни изчисления, които могат да се изпълнят от нея, „изпълнението” се делегира на машината. По-точно описанието на дейността на програмиста е като „разработка на клас изчисления” а не „изготвяне на програми”.

Програмата никога не е самоцел, програмата предизвиква изчисления, а в резултат от тях се получава нужния резултат. Независимо от това, че програмата е крайният продукт на програмиста, „възможните изчисления”, „осъществявани” от машината – ето това е истинското съдържание на неговия труд. Сложността на ситуацията се задълбочава от обстоятелството, че последният етап на този процес, т.е. прехода от (статична) програма към (динамично) изчисление всъщност се предоставя на машината. Затова в някакъв смисъл създаването на програма представлява по-голяма трудност от разработка на математическа теория: и програмата и теорията са структурни, независещи от времето обекти, но ако математическата теория има непосредствен смисъл, то програмата придобива смисъл само по време на изпълнението ѝ [4].

Програмирането става толкова сложно, че придобиват относителна важност и други аспекти като лекотата, с която можем да разберем програмата, можем да се убедим в нейната коректност или можем да я модифицираме и т.н.

“Да се обучи новото поколение програмисти с понижен праг на търпимост към сложността и да ги научим да търсят наистина простото решение това е един от най-важните проблеми в нашата област. Как да убедим хората, че в програмирането простотата и яснотата – накратко: това е, което математиците наричат елегантност – това не е излишен разкош, а жизнено важен въпрос, който определя избора между успеха и провала” [1].

Това мнение се подкрепя и от Н.Уирт “...самоцелната сложност поражда множество проблеми. Главният, тя размива разликата между това, което е действително важно и това, което е ефимерно. Бедата е в това, че се изисква много повече талант, проницателност и време за проектиране на икономична, проста и ефективна система, отколкото сложна и тромава” [6].

Не трябва да се забравя, че има разлика между програми, използвани за обучение и практически използвани програми. Преподавателят пише програми с цел обучение, т.е. те трябва да са лесни за разбиране. Иначе защо ще ги пазим в текстов вид? Ето защо проблемът за стила е от съществена важност.

Стилът на програмиране може да спомогне за подобряване на резултатите от обучението по програмиране. Личният ми преподавателски

опит показва, че е доста по-лесно да се формират умения за добър стил у студентите, които нямат предварителен програмистки опит и безнадеждно трудно да накараш обучаемите на приемат добър стил на програмиране, след като те са писали програми с лош стил в продължение на няколко години [7].

В началното обучение по програмиране, ако се наложи избор между стил и ефективност трябва да се предпочете стила, защото стилната (ясна) програма, когато е осмислена, може да се доработи до ефективна, което често води до загуба на яснота и прави трудно разбирането за новака. Работим с малки програми, избягваме излишната сложност, и се надяваме за възможна методологическа екстраполация за големи задачи.

За студенти по математика и информатика (или информатика с педагогическа правоспособност) ползата се удвоява с готовност за обучение в стил, т.е. от една страна, когато ги обучаваме това дава по-добри резултати, а от друга в бъдеще те са готови също да обучават в стил. Стилът на програмиране в обучението има много важна роля за специалистите, защото те трябва да се научат да пишат ефективни от гледна точка на време и ресурси програми, но и програми с ясни идеи, позволяващи работа в колектив и лесна модификация.

Преглеждайки публикациите по темата прави впечатление, че основните бележки по стила се изразяват в записа на програмния код. Свеждането на понятието стил на програмиране само до правилата за запис на текста на програмата би било непълно**.** Тези стилови особености имат препоръчителен характер. Оформянето на програмните текстове зависи донякъде от езика за програмиране.

Нашият интерес е концентриран най-вече върху онези елементи на стила, които влияят върху четивността, ефективността и разбира се верността на програмата.

4. Стил в началото. Стилът, към който се придържа програмиста, се проявява по време на работата на програмата. Въпросите как са написани програмите, ефективни ли са, лесно ли се четат и разбират, остават като частни, адресирани към малка част от обучаемите. Рядко се коментират съдържателни елементи на стила. Или, ако това се прави, е в края на курса. Може би в този подход има логика. Първо да изучим инструмента, а след това да овладяваме майсторството. Това обаче ни води към нов проблем.

Пренаучаването винаги е много по-трудно, отколкото да се научиш на правилната техника от самото начало, въпреки че „правилното обучение може да е доста скучна работа”. Както пише в [1] Дейкстра „... моята основна цел при обучението по програмиране е да науча студентите отначало да мислят, а не да бързат да се хвърлят в кодиране...” Видимо е, че начинаещите програмисти трябва достатъчно дълго да се обучават “с молив и лист” на

елементи на логиката, правилно построяване на цикъл и т. н. преди да ги пуснем пред клавиатурата със задание да напишат работеща програма. Поради това обучението в стил трябва да върви паралелно с усвояване на алгоритмите и езика за програмиране. Това прави много важен избора на първи език за програмиране.

Стилът на програмиране не се свежда до доброто знание на конкретен език за програмиране, знанието на неговите възможности и правила за запис на програмата независимо, че всичко това се предполага. Лошо е, ако обучаемите подценяват значението на програмната яснота и третират критерия само като специфични правила на езика [9].

Убедителността на резултата силно зависи от яснотата на програмата, от степента, в която тя отразява структурата на изпълнимия процес. Ударението трябва да падне върху уменията за решаване на задачи, а не изучаване на особеностите на езика за програмиране. Искаме да насочим нашето мислене, да му предадем някаква интелектуална дисциплина, защото така може да се избегнат ненужни усложнения в работата.

По-скоро тези знания ще помогнат да се различи програма, написана в добър стил, от програма, написана в лош стил. Има огромна разлика между това да знаеш един език и да владееш този език. Но даже да овладееш един език до съвършенство това е само незначително приближаване към овладяване на стила на програмиране.

Изборът на първи език за програмиране и стила на програмиране имат помощна, но не и маловажна роля в обучението по програмиране. В този контекст се повишават изискванията и към инструмента, който ще се използва.

Подкрепа в тези притеснения се изразяват и в [5]. Обучение в съвременни ефективни методи на програмисти с вече изградени навици е почти невъзможно, защото ги влече непрекъснато към негодните инструменти.

На нашето формиране голямо влияние оказват инструментите, с които работим, в частност формализмите, които използваме, формират начина ни на мислене в положителна или отрицателна посока, а това значи да сме много внимателни в избора как да учим и какво да учим, защото после да се отучим е съвършено невъзможно [1].

5. Заключение. Възможността да се напише хубава и елегантна програми е вече много важно умение, което не винаги е във фокуса на компютърната наука и курсове за програмиране. Има едно добро наблюдение направено от Керниган и Пайк [9] за стила на програмата:

“В свят... на непрекъснат натиск за повече от всичко, единственото което можем да пропуснем са основните принципи – простота, яснота и общност – които формират здравата основа на добрия software.”

Има разлика между стил на преподаване и стил на програмиране. Всеки преподавател има свой собствен стил. Стилът на програмиране представлява съдържателната компонента на стила на преподаване.

Стилът на програмиране е важна характеристика на добрата програма. Статистическият анализ показва, че съществува пряка зависимост между точността на представяне на програмата и стила на програмиране на олимпиадите по информатика. Има полза от стила на програмиране, както за добрите обучаеми (състезатели) – ефективността, разбиването на подзадачи, така и за начинаещите – подреждане, оформяне, яснота и др. В обучението програмите с различно качество на стила на програмиране не трябва да бъдат третирани еднакво. Отговорното отношение към стила на програмиране е много важно и за състезателите в началните им стъпки [7].

Литература:

[1]. Дейкстра Е, Почему программное обеспечение такое дорогое? http://club.shelek.ru/view.php?id=32

[2]. Дейкстра Е, Почему американская информатика кажется неизлечимой, http://club.shelek.ru/view.php?id=32

[3]. Дейкстра Е, Ответы на вопросы студентов отделения программного обеспечения, http://club.shelek.ru/view.php?id=32

[4]. Дал У., Дейкстра Э., Хоор К. Структурное программирование. – М.:”Мир”, 1975. – с.7 – 97.

[5]. Система образования как фактор национального суверенитета в сфере информационньх техологий, Ф.В. Ткачов, доклад представлен на Совещание... ,10,2006

[6]. Wirth N, Computing Science Education: The Road not Taken, ITiCSE Conference, Aarhus, Denmark, 2002, http://www.inr.ac.ru/~info21/texts/2002-06-Aarhus/en.htm

[7]. Skūpienė J., Programming Style – Part of Grading Scheme in Informatics Olympiads: Lithuanian Experience, ISSEP,2006,p. 545-552

[8]. Grigas, G., Investigation of the relationship between program correctness and programming style, Informatica, vol. 6, No. 3 (1995) 265–276

[9]. Kernighan, B. W., Pike, R.: The Practice of Programming. Addison-Wesley (1999)

[10]. Mohan, A., Gold, N.: Programming Style Changes in Evolving Source Code. In: IEEE Proceedings of the 12th International Workshop on Program Comprehension, Bari, Italy, June 24–26, 2004 (2004) 236–240