РОЛЯТА НА ЕЗИКА В УВОДНОТО ОБУЧЕНИЕ ПО ПРОГРАМИРАНЕ

Бойко Банчев

Институт по математика и информатика — Българска академия на науките
boykobb@gmail.com

***Резюме**: Изхождайки от съвременен възглед за обучението по програмиране се формулират желателни свойства на език, подходящ за уводно обучение. На основата на програми, решаващи няколко най-прости задачи, се прави сравнителна оценка на Java и Ruby в тази роля. Изтъква се значението на онагледяването в обучението по програмиране и се посочва сполучливо решение за графично онагледяване.*

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

1. Увод

Ефективността на обучението по програмиране – особено когато става дума за въвеждане в тази дисциплина – съществено зависи от използвания език за програмиране. Влиянието на езика е в две посоки. От една страна, от него очакваме да благоприятства разбирането и усвояването на необходимите понятия и връзки в информатиката и програмирането, да подпомага и да прави удобно онагледяването им с конкретни програми и да насърчава експериментирането и изследването, съпътстващи тази дейност. От друга страна, използването на конкретен език, а и среда за програмиране на него, налага опознаване и съобразяване с множество понятия, правила и съглашения, характерни за езика и изпълнителната среда, но не непременно отнасящи се до същината на програмирането, а понякога и спъващи обучението. Това обуславя необходимостта от добре премислен избор на език за програмиране в началното обучение по него, както и значимостта на последствията от този избор за успеха и качеството на обучението.

Казаното, с някои уговорки и ред необходими уточнения и допълнения, се отнася до всички разновидности на уводното обучение по програмиране – училищно, университетско или друго, предназначено за една или друга възраст, с каквато и да е степен на задълбоченост. Затова тук се спираме на общи въпроси, пряко или косвено отнасящи се до избора на език за програмиране, без оглед на особеностите, свойствени на различните обучителни контексти.

2. Цел и съдържание на обучението по програмиране

Разглеждането на ролята на езика за програмиране в обучението следва да се основава на ясен възглед за целта и съдържанието на самото обучение.

Езиците за програмиране са най-прякото и значимо въплъщение на представите ни за програмирането. Тези представи се изменят с времето, което води до еволюция на езиците – съществуващите се променят, появяват се нови. Налице е и обратното влияние: широкото използване на даден език или тип езици в обучението и практиката на програмирането оказва определящо влияние върху формирането и преобладаването на понятийни представи и връзки, на схеми и навици на мислене в тази високо абстрактна област, върху развиването на едни и потискането на други системи от понятия и парадигми на програмирането.

Това подчертава както значимостта на избора на език за обучение, така и нуждата от надезиков подход, произтичащ от възгледа ни за целта и съдържанието на обучението по програмиране. Затова преди всичко да се опитаме да направим обзор именно на този възглед.

Според нас целта на обучението по програмиране е да дава:

- знания за алгоритмични и даннови структури,
- за принципи и подходи при изграждането им и
- умения тези знания да се прилагат при решаване на задачи чрез програмиране.

Постигането на тази практическа цел минава през усвояване на фундаментални, дълбинни категории на информатиката като действие, достъп, именуване, цитиране, косвеност, локалност, следване, агрегиране, подчиняване, разклоняване, повторност и др. Значителна роля в този процес има развиването на способност за абстрахиране и структурно декомпозиране – тъкмо тя е в основата на аналитичното мислене от характерния за информатиката вид.

В прагматичен план задача на обучението по програмиране е развиването на умения за програмиране на един или повече езика на основата на фундаментални познания в информатиката.

Като боравещо с информация и същевременно обвързано с конкретни езикови средства за изразяване, програмирането и съответно обучението по него включват, в най-общ план, знания от следните области:

- основни видове информация;
- основни форми на структуриране на информация;
- свойства на различните видове информация и информационни структури и действия с тях;
- концептуални и езикови средства за изграждане на програми.

Сред основните видове информация (без да търсим изчерпателност) можем да посочим следните, наред с най-характерните за тях действия.

Текст

— Пораждане и анализ на думи, редове и други структури.

— Образуване на текстово представяне на число и обратно.

— Разпознаване чрез образци (типично чрез т. нар. регулярни изрази).

Числа и прости числови структури (координатни двойки, дроби, полиноми)

— Пресмятания с такъв вид информация.

Булеви (логически) стойности

— Формиране, преобразуване и пресмятане на логически изрази.

Символи (не литери (character data), а това, което думата обозначава в езиците Lisp, Prolog, Erlang, Ruby и др.)

— Пораждане и различаване.

Специфични: час, дата, име на файл и др.

— Преобразуване в различни форми, вкл. към и от текст, разчленяване, намиране на времеви разлики и други отношения.

Колкото до формите на структуриране на информация, преобладаваща употреба имат следните разновидности:

- линейни, разклонени (или вариантни) и йерархични;
- последователни и с явни връзки: едномерни (редици), двумерни (правоъгълни и др. таблици) и множество други;
- хомогенни и нехомогенни;
- асоциативни таблици;
- множества – пряко или косвено представени.

Повечето структурни схеми могат да се отнасят както до даннови, така и до управленски структури.

Накрая, но разбира се с особена важност, са алгоритмичните познания и умението те да се прилагат. Главни класове алгоритми са тези за изброяване, търсене и подреждане на разнообразни структури от данни. Следва да изтъкнем сравнително скорошното в информатиката прозрение, че и трите осъществяват съдържателна връзка на какъв да е вид структура с понятието редица – факт, който подчертава принципната и практическа изключителност на редиците сред другите форми на структурност.

3. Език за програмиране и изпълнителна среда

Твърде често обучението по програмиране се свежда до борба със синтактичните и други чудатости на някой език за програмиране. Решаването дори на много прости задачи се препъва в присъщите на езика особености без връзка с тези задачи и изобщо със същността на програмирането.

Това явление в крайна сметка води до израждане и на самите програми за обучение: вместо върху описаното по-горе съдържание на програмирането, учебните програми се съсредоточават над елементи на езика като условни и повтарящи команди, масиви, процедури и пр. Вместо начини за организиране и полезни действия с данни, такива учебни програми натрапват запознанство

само с наличните в даден език изразни средства. Фактически се подменя съдържанието на обучението по програмиране и то се профанира.

Описаният недостатък трябва да бъде осъзнат и може да бъде преодолян. Последното предполага използване в обучението по програмиране на адекватен език – такъв, който притежава подходящи за тази роля свойства. Смятаме, че такива са следните:

- изразителност, включително наличие на съвременни и перспективни изразни средства;
- простота, непосредственост на изразяването – минимум бюрократизъм, синтактични особености и други пречки;
- никаква или много ниска йерархичност на стандартната библиотека (йерархичността в този случай затруднява търсенето и използването);
- универсална приложимост, широка достъпност и практическа използваемост.

Особено важни сред изразните средства на езика са:

- разнообразни, обобщени по тип структури от данни и алгоритми (множество готови);
- действия над структурни данни (редици, множества и др.) като цяло;
- елементи на функционално програмиране – стойности-функции (безименни функции) в една или друга форма, композиране на функции, частично прилагане;
- средства за анализ и преобразуване на текст;
- възможност за несложно моделиране на даннови и управленски структури, недадени наготово в езика.

Важна страна на процеса на обучение по програмиране е насърчаването към използване на подходящи, характерни за езика и стандартната библиотека към него средства, без при това изучаването на последните да се превръща в самоцел.

Много съществена за ефективността на обучението по програмиране е работната среда, в която се създават и изпълняват програмите на избрания език. Интерактивните среди, характерни за т. нар. динамични езици, имат огромни предимства в това отношение. Основното сред тях е възможността да се изпълняват какви да е програмни действия, дори най-малки, например да се пресмятат изрази или да се задават имена на стойности. Съответният програмен текст не е нужно да се извлича от файл или другаде, а най-често се създава там, където се изпълнява. Така всяко програмно действие се съпровожда възможно най-непосредствено от полезна обратна връзка.

В интерактивна среда отсъстват или незадължително се минава през преобразования на програмата като компилиране и свързване. Спестяването на

такива и други действия опростява взаимодействието на обучаемия със средата и така увеличава достъпността и във всякакъв смисъл. Действия с файлове като носители на програмата няма или са много малко, което също благоприятства съсредоточаването върху програмирането по същество.

Като правило, с интерактивна среда за изпълнение сред съвременните езици биват снабдени тези, които наричаме динамични. Тези езици – например Smalltalk, Ruby, Python, JavaScript, Scheme, Lua и Julia – имат и ред други черти, които ги правят особено подходящи за въвеждане в програмирането: те са изразителни, небюрократични и благоприятстват кратко изразяване. С оглед на редица вторични фактори, които би било отегчително да изброяваме тук, сред динамичните езици за най-привлекателни за начално обучение по програмиране смятаме Ruby и Python и по-специално първия от тях.

Освен че има посочените по-горе желателни качества, Ruby се отличава със следните черти:

- краткост, простота и нагледност на записа;
- авторова философия, съчетаваща прагматичност и естетика;
- неограничено големи цели и дробни числа по подразбиране (няма препълване или загуба на точност);
- поддържане на няколко основни стила (парадигми) на програмиране: процедурен, обектноориентиран, отчасти функционален;
- възможност за (пре)определяне на операции;
- рефлексивност и възможности за метапрограмиране;
- широко използван в уебпрограмирането, като команден език, за обработване на текстова информация и в други области.

4. Съпоставка на езиците Java и Ruby чрез примери

Езикът Java, макар неоправдано и вече по-рядко, бива използван за начално обучение по програмиране. Затова да съпоставим решения на три най-прости задачи, каквито обичайно се разглеждат в самото начало на обучението, на този език и предложения от нас за тази роля Ruby. Всяка от задачите допълва предишната.

По традиция първата програма в обучението по програмиране отпечатва определен текст. На Java най-късият и прост вариант е следният:

```
public class FirstProg {
  public static void main(String []args) {
    System.out.println("Аз програмирам!");
  }
}
```

Програмата съдържа клас с атрибут `public` и главна функция със задължително име `main` и атрибути `public`, `static` и `void`. Последната има задължителен, макар и ненужен параметър – масив `args` от низове. Отпечатването на низа става с действието `println`, чието цитиране изисква да се посочи местонахождението му: обектът `out` от класа `System`. Има и ред синтактични особености: скоби `()`, `[]` и `{}`, знаците `;` и `.` и др. От всичко изброено всъщност само `println` и отпечатваният низ имат отношение към решаваната задача. Останалото е по същество излишно, но не може да се избегне, следователно е белег на тежък езиков бюрократизъм.

Освен това програмата трябва да се намира във файл с име `FirstProg.java` (непременно същото като на класа) и да бъде компилирана, за да се получи друг файл – с изпълним вид на програмата. И в този вид обаче тя се изпълнява не пряко, а само чрез нарочна програма-изпълнител.

Но това е гротеска на програмиране! Невъзможно е в този и вид първата среща с програмирането да не бъде плашеща и отблъскваща.

Програма на Ruby, която прави същото е:

```
puts 'Аз програмирам!'
```

Програмата съдържа само две неща: командата за печатане и извеждания низ. Очевидно тя е възможно най-проста и ясна практически без обяснения.

Следващата програма на Java прочита от един ред текстов низ, колкото се окаже дълъг, и го отпечатва.

```
import java.io.*;
public class ReadString {
  public static void main(String []args)
                                throws IOException {
    BufferedReader in = new BufferedReader(new
                    InputStreamReader(System.in));
    String s = in.readLine();
    System.out.println(s);
  }
}
```

Дори само четенето използва три различни класа, три обекта и един метод (функция) от стандартната библиотека на езика. Два от обектите се създават чрез изрично обръщение към операцията `new`. Главната функция е натоварена с още един атрибут – `throws IOException`, а командата `import` осигурява достъп до използваните класове.

Нищо от това не може да бъде пропуснато! Проблемите за потенциалния обучаем, а и за преподавателя се задълбочават.

Програма на езика Ruby, която прави същото е

```
puts gets
```

и съдържа само по една дума за всяко от действията, четене и писане. Както и по-горе, не е необходимо програмата да се записва някъде – може да се подаде непосредствено на интерпретатора.

Следва програма на Java, която прочита от един ред някакъв (неизвестен предварително, възможно и нулев) брой цели числа и пресмята и отпечатва сбора им. Ако редът не съдържа очакваното, отпечатва се низът `error`.

```
import java.io.*;
import java.util.Scanner;
public class SumIntegers {
  public static void main(String[] args)
                                throws IOException {
    try {
      BufferedReader in = new BufferedReader(new
                  InputStreamReader(System.in));
      Scanner sc = new Scanner(in.readLine());
      int sum = 0;
      while (sc.hasNext())
        sum += sc.nextInt();
      System.out.println(sum);
    }
    catch (Exception e)
      { System.out.println("error"); }
  }
}
```

А програма на Ruby, която прави същото и дори повече е:

```
sum = gets.split.reduce(0){|s,x|s+Integer(x)}
                                    rescue 'error'
puts sum
```

Освен че програмата е много по-кратка от тази на Java, устроена е много по-просто и на практика не съдържа части, които бихме сметнали за неотнасящи се до същността на задачата, тя не е застрашена от препълване, ако пресмятаният сбор се окаже голям. Програмата на Java в този случай не успява да пресметне сбора и печата `error`.

5. Онагледяването в обучението по програмиране

Онагледяване се използва, за да се покажат чрез текст, включително във вид на таблица или схема, или чрез рисунка съдържанието и промените на данни, както и резултата от работата на програма. Подразбираме онагледяване

по програмен път. В най-простия случай то се свежда до непосредствено отпечатване на стойности, но са потребни и по-развити форми.

В редица случаи, според вида и сложността на данните, помощта на онагледяването в програмирането е безценна и това важи с особена сила в обучението. Да си представим колко време би отнело да преценим по три двойки координати дали триъгълникът с тези върхове е тъпо-, остро- или правоъгълен. А колко би отнело да разберем по координатите на върховете и дали дадена 20-звенна начупена линия се самопресича? С помощта на автоматично породен чертеж на тези въпроси се отговаря за миг.

Онагледяването, в частност графичното:

- ефикасно подпомага разбирането на структури и процеси;
- е привлекателно за обучаемите;
- трябва да бъде достъпно за тях;
- не бива да бъде самоцел;
- трябва във възможно най-малка степен да принуждава към допълнителни усилия и използване на специфични езикови средства.

Във връзка с графичното онагледяване обаче съществува проблем и той отново е езиков: графичните средства в езиците за програмиране или са специфично обвързани с операционна среда и твърде сложни за употреба, или изобщо отсъстват. Ето как изглежда възможно най-простата програма за рисуване на отсечка на езика Java:

```
import java.awt.*;
import javax.swing.*;

public class Drawline extends JPanel {
  public void paintComponent(Graphics g) {
    g.drawLine(10,10,100,50);
  }

  public static void main(String[] args) {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation
                          (JFrame.EXIT_ON_CLOSE);
    frame.setSize(200,100);
    Drawline panel = new Drawline();
    frame.add(panel);
    frame.setVisible(true);
  }
}
```

Програмата съдържа редица подробности, които не могат да се заобиколят, а всъщност нямат никакво отношение към чертането на отсечка. Абсурдно е толкова сложно средство да се усвои и прилага от начинаещи обучаеми по програмиране. Програмата на Ruby за решаване на същата проста задача в този случай е отново по-кратка и донякъде по-проста, но не наистина проста:

```
require 'tk'
TkRoot.new do |root|
  TkCanvas.new(root) do |canvas|
    pack('side'=>'top')
    TkcLine.new(canvas,[10,10,100,50])
  end
end
Tk.mainloop
```

Усвояване на графични средства като показаните на практика изисква овладяване на отделен (под)език за програмиране – значително усилие, което е неоправдано в уводен или друг общ курс по програмиране.

Графично онагледяване, свободно от посочения недостатък, може да се постигне, като се използва отделно, независещо от езика за програмиране и операционната система графопораждащо средство или няколко такива. Едно такова средство е `sp`, създадено от автора именно с такава цел.

`sp` е език за описване на равнинни фигури и транслатор за него. Отличителните му черти са следните:

- образите са векторни (мащабируеми, получавани чрез геометрични построения);
- координатната система е права, а единиците са мм или кратни на мм, а не екранни пиксели (в примерите на Java и Ruby по-горе е обратно);
- входният език е много прост (овладява се за минути), но достатъчно изразителен и текстът на него лесно се поражда по автоматичен път (чрез програма на кой да е език);
- резултатът е SVG или EPS (оттам и PDF) – съответно за уеб и печатно публикуване и е видим на всеки компютър с обичайните широко разпространени средства (напр. SVG – чрез браузър);
- езикът е подробно описан, а програмата-транслатор е свободно достъпна и използваема във всички разпространени операционни системи.

За сравнение с примерите на Java и Ruby по-горе, входният текст за `sp` за рисуване на отсечка е следният:

```
lines
10 10 100 50
```

Литература

1. Bantchev B. sp: user manual and reference. http://www.math.bas.bg/bantchev/sp.
2. Oracle Technology Network – Java. http://oracle.com/technetwork/java.
3. The Ruby programming language home page. http://ruby-lang.org.

THE ROLE OF THE LANGUAGE IN TEACHING INTRODUCTORY PROGRAMMING

Boyko Bantchev

Institute of mathematics and informatics — Bulgarian academy of sciences
boykobb@gmail.com

***Abstract**: Based on a present-day view of teaching programming, desirable features are formulated for a language suitable for teaching introductory programming. On the examples of programs solving several very simple problems, a comparative evaluation is done of the languages Java and Ruby in this respect. The significance of visualization in programming education is outlined, and a suitable solution for graphical visualization is pointed at.*

***Keywords**: programming language, education.*