Езикът за уводно обучение по програмиране и около него

Бойко Банчев

boykobb@gmail.com

Институт по математика и информатика — БАН

октомври 2017

Цел и съдържание на обучението по програмиране

Цел и предпоставки

Според мен 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* – тя е в основата на аналитичното мислене от характерния за информатиката вид.

Програмиране и информатика

Информатиката не е само програмиране, но програмирането, разбирано в широк смисъл, е нейна сърцевина.

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

Съдържание: общ план

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

Основни видове информация и действия с тях

Текст

- пораждане и анализ на текстова информация (думи, числа, редове и др.);
- образуване на текстово представяне на число и обратно;
- разпознаване чрез образци (типично чрез т. нар. регулярни изрази)

Числа и прости числови структури

(координатни двойки и тройки, дроби, полиноми)

- пресмятания с такъв вид информация

Булеви (логически) стойности

- формиране, преобразуване и пресмятане на условия (логически изрази)

Символи

(Символите (англ. symbolic data) не са литери (character data))

- формиране 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 др.

Основни видове структури

Линейни / разклонени (вариантни) / йерархични

Последователни (естествена подредба) / с явни връзки

Едномерни (редици), двумерни (правоъгълни и др. таблици) и редица други (т. нар. „масиви“ са само частен и не твърде сполучлив пример за редици)

Хомогенни / нехомогенни

Асоциативни таблици

Множества (пряко или косвено представени)

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

Три фундаментални типа алгоритми над структурни данни:

изброяване, **търсене** и **подреждане**.

И трите осъществяват съдържателна връзка на какъв да е вид структура с понятието редица (сравнително скорошно прозрение в информатиката).

Език за програмиране, изпълнителна среда и други фактори

Подмяна на съдържанието на програмирането

Твърде често обучението по програмиране се свежда до борба със синтактичните и други чудатости на някой език за програмиране. Моделирането дори на прости понятия и структури се препъва в многобройните наложени от езика присъщи нему особености, нямащи нищо общо нито с решаваните задачи, нито изобщо със същността на програмирането.

Това явление в крайна сметка води до израждане и на самите програми за обучение: вместо върху основите на програмирането или похватите и методите, характерни за някоя негова област, програмите се съсредоточават върху съставните части на езика – условни команди, команди за цикъл, масиви, процедури и др. Вместо начини за организиране на данни и полезни действия с тях, такива програми предлагат запознанство само с наличните в някой език изразни средства. В този смисъл те са безполезни.

За сравнение

В курс по готварство опознаваме видовете храна и начините за приготвянето ѝ, а не инструментите – тенджери, черпаци и печки.

В курс по боядисване научаваме какви видове боядисване има, какви методи и похвати се прилагат в тази дейност, какви предварителни или следващи действия са нужни при боядисване и само във връзка с тези знания, подчинено на тях се обсъжда какви материали и инструменти, напр. бои, четки и съдове се използват, защо има различни видове от тях и кой вид за какво и как се употребява.

В курс по готварство се изучава приготвянето на храна, а в курс по бояджийство – боядисването.
Но в учебните програми по програмиране често няма програмиране!

Какъв ЕП за (начално) обучение?

Описаният недостатък – впрочем съвсем не рядко срещан – трябва да бъде осъзнат и може да бъде преодолян. Последното е свързано със сполучливото избиране на език за обучение по програмиране, което пък предполага наличие у езика на подходящи за тази роля свойства.

Смятам, че такива са следните:

- **изразителност**, вкл. 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 (йерархията натрапва зависимости и в този смисъл вреди на простотата – отдавна забелязано!);
- **универсална приложимост**, широка достъпност и практическа използваемост (вкл. и в перспектива).

Изразни средства на ЕП

Особено важни сред изразните средства на езика са:

- разнообразни, обобщени по тип структури от данни и алгоритми (множество готови);
- действия със структурни данни (редици, множества и др.) като цяло, вместо само с частите им;
- 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.

Изпълнителна среда

Много съществена за ефективността на обучението по програмиране е изпълнителната среда, в която се (създават и) изпълняват програмите на него.

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 фрагменти, вкл. и най-малки, например пресмятане на изрази или задаване на имена на стойности. Самите фрагменти не е нужно да се извличат от файл или другаде, а най-често се създават там, където се изпълняват. Така изпълняването на каквото и да е се съпровожда възможно най-непосредствено от необходимата обратна връзка.

Отсъстват или незадължително се минава през преобразования на програмата като компилиране и свързване. Спестяването на такива и други действия опростява взаимодействието на обучаемия с изпълнителната среда и така увеличава достъпността ѝ във всякакъв смисъл.

Действията с файлове като носители на програмата отсъстват или са сведени до минимум, с което се премахва още един слой от неотнасящи се пряко до създаването на програма действия.

Избор на ЕП – някои (привидни) възможности

(За повече езици виж напр. http://www.math.bas.bg/bantchev/place)

- C – изключително широко използван, сравнително прост, но липсват съвременни изразни средства; задължителен за професионални информатици, но не твърде подходящ като уводен
- C++ – изразителен, универсален, но донякъде бюрократичен; много подходящ напр. за обучение по структури от данни, алгоритми и др. под. на относително напреднали, но недостатъчно подходящ за уводен в програмирането
- Java – много широко използван, но изключително бюрократичен; особено неподходящ за уводен в програмирането
- C# – почти толкова тромав като Java, а и достъпен само чрез Microsoft .NET
- D – модерен, изразителен, универсален и сравнително прост за изучаване, но не много широко разпространен
- Go (Google), Rust (Mozilla), Swift (Apple) – модерни, универсални и перспективни езици от съответните фирми, но засега не много популярни у нас

Динамичните ЕП

Динамичните ЕП са като цяло особено подходящи за въвеждане в програмирането, тъй като са изразителни, небюрократични, благоприятстват кратко изразяване и предлагат или допускат изпълняване в интерактивна среда.

Такива езици са например, сред много други, `Smalltalk`, `Perl`, `Ruby`, `Python`, `JavaScript`, `Scheme`, `Lua` и `Julia`.

С оглед на редица фактори (виж напр. по-горе), сред динамичните езици за най-привлекателни за начално обучение по програмиране смятам `Ruby` и `Python` и по-специално първия от тях.

Езикът Ruby

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

- авторова философия – производителност и удоволствие от програмирането;
- краткост, простота и нагледност на записа;
- неограничено големи цели и дробни числа по подразбиране (няма препълване или загуба на точност);
- поддържане на няколко основни стила (парадигми) на програмиране: процедурен, ОО, отчасти функционален;
- гъвкав ОО модел;
- възможност за (пре)определяне на операции;
- съпрограмност;
- рефлексивност и възможности за метапрограмиране;
- широко използван в уебпрограмирането, като команден език на ОС, за обработване на текстова информация и в др. области.

Съпоставяне на 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 Java и, като възможна алтернатива, Ruby.

Всяка задача допълва предишната.

Обичайната първа програма: Java

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 не се изпълнява пряко, а с помощта на друга, нарочна програма-изпълнител.

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

Програма на Ruby, която прави същото

```
puts 'Аз програмирам!'
```

или дори само (пряко в интерактивната среда)

```
'Аз програмирам!'
```

Програмата съдържа само две неща: командата за печатане и низа (или само второто). Очевидно тя е възможно най-проста и ясна практически без обяснения.

Програма за четене на низ: Java

Програмата 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);
  }
}
```

Четенето става по каноничния според определението на езика начин. Дори само то 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.

Всичко изброено е необходимо, не може да бъде пропуснато!
Гротеската се задълбочава.

Програма на Ruby, която прави същото

```
puts gets
```

(Както и по-горе, не е необходимо програмата да се записва някъде – може да се подаде непосредствено на интерпретатора)

Сумиране на редица от числа: Java

Прочита се от един ред някакъв (неизвестен предварително, възможно и нулев) брой цели числа и се пресмята и отпечатва сборът 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"); }
  }
}
```

Програма на Ruby, която прави същото (и дори повече)

```
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 съществото на задачата, тя не е застрашена от препълване, ако някое число или пресмятаният сбор се окажат големи. Програмата на Java в този случай не успява да пресметне сбора и печата `error`.

JavaScript – все по-привлекателна възможност

Браузерът като среда за пресмятане и програмиране

`[5,-2,7,3].reverse()`

Обръщане на редица

`[5,-2,7,3].reduce((a,b)=>a+b)`

Сбор на числа от редица

`[5,-2,7,3].reduce((a,b)=>a*b)`

Произведение

```
poly = (c,x) => c.reduce((a,b)=>a*x+b)
```

Функция за пресмятане на полином от една променлива

`poly([2,3,-5,1],2)`

Пресмятане на полинома $2x^3+3x^2-5x+1$ за $x=2$

`[1,-2,5,3].map((x)=>poly([2,3,-5,1],x))`

Пресмятане на същия полином наведнъж за $x=1,-2,5,3$

За онагледяването в обучението по програмиране

Мястото на онагледяването. Езикови средства

Онагледяване се използва, за да се покажат чрез текст, вкл. във вид на таблица или схема, или чрез рисунка съдържанието и промените на данни, както и резултата от работата на програма. В най-простия случай то се свежда до непосредствено отпечатване на стойности, но има смисъл да се използват и по-развити форми.

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.

Проблем: графичните средства в езиците за програмиране или са специфично обвързани с операционна среда и сложни за употреба, или изобщо отсъстват.

Пример: програма за рисуване на отсечка на 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);
  }
}
```

Програма на Ruby, която прави същото

```
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
```

И в двата показани примера програмите са най-малките и най-простите възможни!

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.

Решение на проблема: да се използва отделно, независещо от ЕП и ОС графопораждащо средство или няколко такива.

Едно такова средство е `sp`.

sp: език и транслатор за равнинни фигури

Основни свойства на sp:

- 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 ЕП);
- резултатът е SVG или EPS (оттам и PDF) – съответно за уеб и печатно публикуване и видим на всеки компютър без нужда от специални средства;
- програмата-транслатор е достъпна за всяка ОС.

Рисуване на отсечка на sp (срв. с Java и Ruby по-горе)

```
lines
10 10 100 50
```

Някои програмно породени с помощта на sp образи