Как «встроить» программную инженерию в курсы по математике для программистов?

Николай Вячеславович Шилов

(по материалам совместной работы «Лабораторные работы по математике для студентов-программистов» с И.В. Конюховым, Я.А. Холодовым, З.Р. Ягудиным и С.В. Масягиным)

АНО ВО «Университет Иннополис», Иннополис

Что мы хотим обсудить?

- Мы хотим представить опыт преподавания математических дисциплин в Университете Иннополис, где студенты получают специальность «Информатика и вычислительная техника», как-то
  - «внесение» повести и методов Программной Инженерии в математические курсы,
  - объединение обязательных лабораторных работ по математическим дисциплинам с профильными предметами, связанными с разработкой программного обеспечения,
  - опциональные лабораторные работы, призванные популяризировать как математику, так и искусство программирования.

Формальная спецификация и верификация алгоритмов вычислительных методов в курсе «Дифференциальные Уравнения»

Часть 1

What is the *standard* `sqrt`?

- Specification says (*C reference* at http://en.cppreference.com/w/c/numeric/math/sqrt):

  “sqrt, sqrtf, sqrtl

  C Numerics Common mathematical functions

  Defined in header <math.h>

  …

  Parameters

  arg - floating point value

  Return value

  If no errors occur, square root of arg , is returned.”

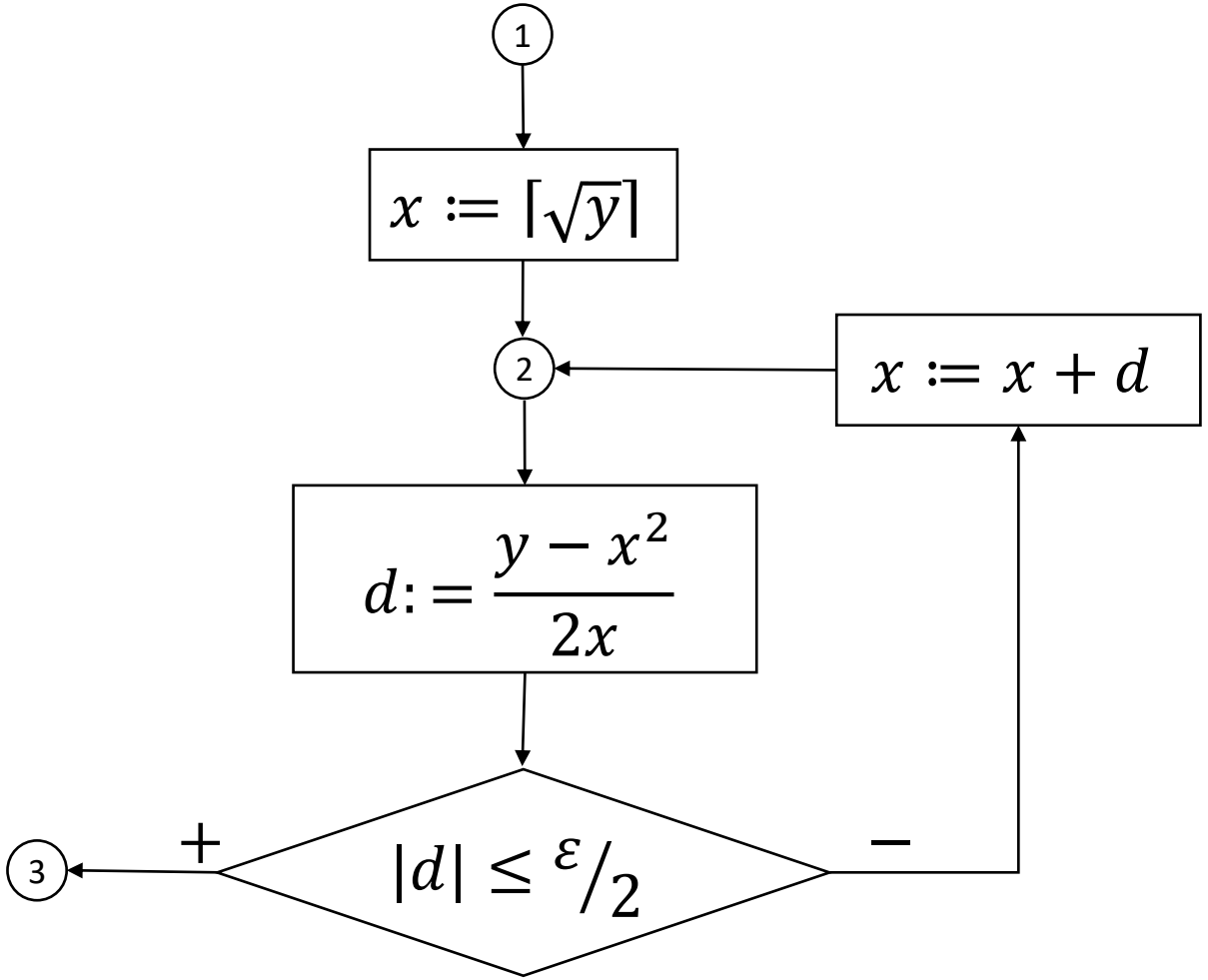
Alternatives for the standard `sqrt`

- It makes sense to introduce another function with two arguments $SQRT(y, \varepsilon)$ where y stays for the argument and ε stays for accuracy (precision), that can be mathematically specified by the following clauses:
  - if $y \geq 0$ and $\varepsilon > 0$ then $SQR(y, \varepsilon)$ must return a (real) value $x \geq 0$ that differs from $\sqrt{y}$ not more than ε, i.e., *truncation error is* $|x - \sqrt{y}| \leq \varepsilon$.

How to compute $SQRT$ using Newton-Raphson method

- Expected input: Any non-negative real value $y \geq 0$.
- Method
  - Let x_0 be initial approximation;
  - if x_m ($m \geq 0$) is defined already , then
    - let $d_m = \frac{y - {x_m}^2}{2x_m}$ (intuition: $y = (x_m + \varepsilon_m)^2 \approx x_m^2 + 2x_m\varepsilon_m$);
    - and $x_{m+1} = x_m + d_m$ (intuition: $x_{m+1} = \frac{x_m + {y}/{x_m}}{2}$);
  - pick-up some x_n ($n \geq 0$) that has an appropriate accuracy.
- Expected output: A real value x_n approximating $\sqrt{y}$.

$SQRT$ algorithm (in precise arithmetic)



А.П. Ершов – С.С. Лаврову (март 1983 г.)

Поздравляю с днем рождения, милый друг!
Шестьдесят уже как выпало на круг.
Завтра то ли, как вчера, себя вести,
То ли новую программу завести.

Чтобы грамотно программу составлять,
Надо пред- и постусловия задать,
Надо точный счетчик времени иметь
И циклический инвариант привлечь.

Предусловие — твой праздничный венец,
Постусловие — у всех один конец,
Время катится без помощи чужой,
Для инварианта — будь самим собой!

$SQRT$ algorithm: specification, annotations and verification

- Specification:
  - precondition: $y > 1$(!) and $\varepsilon > 0$;
  - postcondition: $|x - \sqrt{y}| \leq \varepsilon$;
  - Hoare triple: $[y > 1 \ \& \ \varepsilon > 0]SQRT\left[|x - \sqrt{y}| \leq \varepsilon\right]$.
- Annotations (inductive assertions):
  1. precondition;
  2. invariant: precondition altogether with $\sqrt{y} < x$;
  3. postcondition.
- Verification: Floyd-Hoare proof method (for *partial* and *total* correctness).

Exercises

1. Prove *geometric convergence* of the algorithm $SQRT$ i.e., that every next error is (at least) twice less that the previous one; using geometric convergence, evaluate computational complexity of the $SQRT$ algorithm (in terms of number of iterations as a function of input values y and ε).
2. Prove *quadratic convergence* of the algorithm $SQRT$ i.e., that every next error is (at least) square of the previous one; using quadratic convergence, evaluate computational complexity of the $SQRT$ algorithm (in terms of number of iterations as a function of input values y and ε).
3. Suggest a method to compute $\lceil \sqrt{y} \rceil$ (for input $y \geq 0$, $y \in \boldsymbol{R}$) that uses integer variables (but y). Design an algorithm that implements the method, specify, annotate and verify it (using Floyd-Hoare methods). What is computational complexity of the algorithm (in terms of number of iterations as a function of y)?

Обязательные Лабораторные работы в курсах «Линейная алгебра» и «Дифференциальные уравнения»

Часть 2

Курс «Линейная алгебра»

- Курс читается во втором семестре первого года обучения. В первом семестре студенты познакомились с основами аналитической геометрии, понимают основные концепции векторных пространств, умеют пользоваться базовыми приемами решения систем линейных уравнений малой размерности.
- Курс нацелен на углубленное изучение математических основ решения систем линейных уравнений.

Курс «Введение в программирование»

- Курс читается в течение всего первого года обучения. В рамках первого семестра студенты знакомятся с парадигмой объектно-ориентированного программирования на высокоуровневом языке Eiffel.
- Студенты, учатся декомпозировать задачу на отдельные классы, объекты и их отношения. Учащиеся знакомятся с базовыми принципами такого подхода (инкапсуляцией, наследованием, полиморфизмом), особенностями взаимодействия объектов в программе (ассоциацией, композицией, агрегацией).

Курс «Линейная алгебра»

- Большое внимание уделяется эффективности применяемых алгоритмов, оценке их вычислительной сложности, а также особенностям компьютерной реализации. Очевидно, что подобное невозможно без применения современные языков программирования.
- Решение задач в данном курсе не требует написания «больших» программ, что делает код «обозримым» и не требует длительного времени для его разработки.

Курс «Введение в программирование»

- Во втором семестре студенты изучают элементы низкоуровневого программирования на языках C и C++ (как механизма ссылок и указателей, позволяющими тонко управлять процессом распределением памяти).

Первая лабораторная работа по «Линейной алгебре»

- Одной из первых тем курса «Линейная алгебра» является решение систем линейных алгебраических уравнений (СЛАУ) методом Гаусса и Гаусса-Жордана. Математические операции, необходимые для приведения исходной матрицы системы к единичному виду, легко можно записать в форме умножения матриц, используя специальные матрицы исключения, перестановки и масштабирования.
- Задание первой лабораторной работы - программная реализация класса «Прямоугольная матрица» на языке C++ со стандартными математическими операциями сложения, умножения, вычитания, транспонирования, учитывая все особенности таких операций и, при необходимости, генерируя соответствующие исключения.
- Матрицы исключения, перестановки и масштабирования являются производными от единичной матрицы, которая, в свою очередь, является квадратной матрицей. Студенты должны реализовать соответствующие классы «Квадратная матрица», «Единичная матрица», «Матрица исключения», «Матрица перестановки», «Матрица масштабирования», используя механизм наследования.

Вторая и третья лабораторные работы по «Линейной алгебре»

- Задание второй лабораторной работы – используя первую работу, реализовать прямой метод решения СЛАУ методом Гаусса-Жордана, и выполнить декомпозицию матрицы на произведение нижней- и верхней-треугольных матриц.
- Задание третьей лабораторной работы посвящено приближенному решению СЛАУ методом Якоби не в процедурном, а в объектно-ориентированном стиле, при котором необходимо определить новый класс «Вектор» (для которого определить некоторые операции, например, вычисления нормы). – Подобный подход позволяет сформировать устойчивый навык мышления в терминах объектов и отношений, нежели функций и процедур.

Четвертая лабораторная работа по «Линейной алгебре»

- Задание третьей лабораторной работы – реализовать метод наименьших квадратов, который является базовым методом регрессионного анализа, применяющемся для оценки неизвестных параметров регрессионных моделей по выборочным данным. Метод наименьших квадратов основан на минимизации суммы квадратов отклонений некоторых функций от искомых переменных и может использоваться для аппроксимации точечных значений некоторой функции. В данном задании особенно полезной является графическая визуализация полученного результата. С этой целью студентам предлагается использовать библиотеку GNU Plot. На график наносится исходное множество («облако») точек и кривая искомой функции, что позволяет наглядно продемонстрировать смысл метода.

Курс «Дифференциальные уравнения»

- Курс является базовым курсом и читается в течение первого семестре второго года обучения. Он является скорее прикладным курсом, нежели классической математической дисциплиной, и направлен на то, чтобы продемонстрировать, что многие научные задачи сводится к математическим моделям, записанным с использованием дифференциальных уравнений.

Курс «Шаблоны и принципы проектирования»

- Студенты второго курса параллельно посещают курс «Шаблоны и принципы проектирования», где знакомятся с основами взаимодействия объектов в программе (построения взаимосвязей классов, объектов и интерфейсов).
- Курс построен на классической книге Э.Гамма, Дж. Влисидиса, Р. Хелма и Р. Джонсона «Шаблоны проектирования», в которой приводятся различные способы построения архитектуры объектно-ориентированного приложения.

Курс «Дифференциальные уравнения»

- Кроме того, в рамках данного курса вводятся базовые понятия численных методов, такие как сеточная аппроксимация, разностные аналоги производных, а также демонстрируются методы Эйлера и Рунге-Кутта численного решения обыкновенных дифференциальных уравнения первого порядка.
- Методы ООП-проектирования используются в вычислительном практикуме по курсу «Дифференциальные уравнения».

Связь с курсом «Шаблоны и принципы проектирования»

- Важно строго следить за соблюдением принципа единственной обязанности классов, когда каждый класс является отдельной функциональной единицей.
- Принцип подстановки Лисков (абстракция и иерархия данных) можно удачно продемонстрировать при использовании унифицированного вызова конкретных реализаций численных методов решения.
- Также легко можно показать, как применяется принцип открытости/закрытости, когда программные сущности должны быть открыты для расширения, но закрыты для модификации.

Опциональные лабораторные работы к курсам «Математический анализ» и «Дискретная математика»

Часть 3

Курс «Математический анализ»

- Курс читается в течение всего первого года обучения. Лабораторные работы к этому курсу носят опциональный характер, они рассчитаны на добровольцев, готовых выполнить их во время 3-недельной летней учебной практики, но имеют образовательное значение: они призваны помочь студентам понять разницу между «идеальным» вычислительным математическим методом и его «реальным» воплощением в виде компьютерной программы, вызвать интерес к изучению дополнительных тем по курсу и решению интересных программистских задач.
- Приведем примеры тем лабораторных работ к курсу анализа:
  - «точное» (точнее, как можно более точное) вычисление частичных сумм гармонического ряда в границах, представимых типом int;
  - вычисление значений логарифмической функции или тригонометрических функций для аргументов в некотором заданном диапазоне с некоторым шагом.

Пример: лабораторная работа на вычисление косинуса

(тема подсказана Борисом Леонидовичем Файфелем)

- Как известно $\cos x = \sum_{n \in \boldsymbol{N}} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} + \cdots + \frac{x^{4n}}{(4n)!} - \frac{x^{4n+2}}{(4n+2)!} + \cdots$ для любого $x \in \boldsymbol{R}$, причем, (в силу теоремы Лейбница для знакопеременных рядов) сумма остаточных членов по абсолютной величине не превосходит последнего суммированного члена ряда (что в данном случае можно записать как $\left|\sum_{m>n} (-1)^m \frac{x^{2m}}{(2m)!}\right| \leq \frac{x^{2n}}{(2n)!}$ для любого $n \geq 0$).
- Задание лабораторной работы состоит в том, чтобы, используя данное разложение в ряд функции $\cos x$, вычислить ее значения с точностью 0.000001 для аргументов в диапазоне от 0 до 50 радиан с шагом 0.5 радиан.
- Цель лабораторной работы – познакомиться на практике с вычислениями (аналитических) функций на основе разложения в степенной ряд, с возникающими при этом сложности из-за разности между «идеальной» арифметикой вещественных чисел и машинной арифметикой в формате с плавающей запятой и связанных с ней ошибок округления.

Курс «Дискретная математика»

- Курс читается в читается в первом семестре первого года обучения. Главная цель курса этого курса – познакомить студентов с «доказательством», как главным методом науки Математики и ее самыми базовыми понятиями такими как *множества*, *конечная комбинаторика*, *отношения*, *функции*, *графы*, *алгебраические структуры*, *логические формализмы*.
- Программистским «напарниками» (и продолжением) курса «Дискретная математика» выступает курс по «Алгоритмы и структуры данных», который читается во втором семестре первого года обучения (и в котором множество лабораторных работ – особенно по представлению и обходам графов).

Курс «Дискретная математика»

- Поэтому лабораторные работы к курсу «Дискретная математика» в Университете Иннополис носят опциональный характер (так же как лабораторные работы к курсу «Математический анализ»), они рассчитаны на добровольцев, готовых выполнить их во время 3-недельной летней учебной практики. Эти работы призваны помочь студентам познакомиться с дополнительным материалом и заняться решением интересных программистских задач.
- Приведем примеры тем лабораторных работ к курсу дискретной математики:
  - визуализация и строгое математическое объяснение алгоритмических карточных фокусов;
  - реляционная семантика программ над булевскими переменными;
  - экспериментальная проверка законов для «почти всех графов» (например, что почти все графы являются связными).

Пример: лабораторная работа по «проверке» что почти все граф являются связными

- При знакомстве с основами теории графов после введения основных определений, изучается «джентельменский» набор свойств как-то связность, регулярность, полнота, диаметр. Обычно мы рассматриваем какие-то определенные «неслучайные» графы, которые могут лучшим образом проиллюстрировать данные свойства. Но какие свойства будут присущи случайно генерируемым графам? И что такое «случайный» граф?
- Исходя из определения вероятности для конечного пространства, если рассматривать все графы фиксированного порядка n, то случайный граф – это любой из $2^{C_2^n}$ возможных графов, причем, вероятность любого такого графа $1/2^{C_2^n}$. (Будем называть такой подход «равномерным».)
- Однако есть еще одна модель случайного графа, принадлежащая П. Эрдёшу и А. Реньи. Эта модель несколько более сложная и менее интуитивная: помимо порядка графа в ней участвует вероятность p «существования» ребра между двумя любыми вершинами; в частности, если говорить про случайные графы, то нас интересует $p = 0{,}5$.

Пример: лабораторная работа по «проверке» что почти все граф являются связными

- Говорят, что *почти все* графы обладают некоторым свойством, если отношение числа графов порядка n, имеющих это свойство, к числу всех графов этого порядка стремится к 1 при $n \to \infty$, то есть, вероятность того, что случайный граф порядка n, обладает рассматриваемым свойством, стремится к 1 при росте n.

- Оказывается, имеют место следующие совсем свойства почти всех графов:
  1. *почти все графы имеют диаметр 2,*
  2. *почти все графы связны.*

- Здесь уместно заметить, что с «человеческой» точки зрения второе свойство «очевидно», а первое – нет: если попросить группу студентов нарисовать граф, то почти наверняка будет нарисован связный граф, так как *психологически* именно ребра, соединяющие вершины, превращают множество вершин в единый объект «граф»; однако, зачастую люди рисуют не «граф общего вида», но дерево (диаметра больше 2).

Заключение

Выводы

- Представленный нами подход позволяет интегрировать математику и программирование, снижает общую нагрузку на студентов первых годов обучения при выполнении обязательных лабораторных работ, что особенно важно ввиду высокой плотности учебного процесса, когда в одно и то же время студенты осваивают сразу несколько базовых дисциплин.
- Кроме того, важно сразу продемонстрировать студентам, что знания, получаемые в ходе освоения различных предметов, не являются разобщенными, а тесно связаны, когда речь идет о разработке конкретного программного обеспечения для решения тех или иных прикладных задач.
- Однако важно подчеркнуть, что такой подход накладывает определенные квалификационные ограничения на преподавательский состав. Преподаватели математических дисциплин должны хорошо разбираться в основах программирования, в то время как преподаватели программирования должны уметь ответить на профильные вопросы, связанные с реализацией конкретных математических алгоритмов. В идеале на подобных курсах должен работать человек, который может синхронизировать подобный процесс для математических и компьютерных дисциплин.

Использованная литература

1. Непомнящий В.А. (1939- 25 декабря 2021) , Рякин О.М. *Прикладные методы верификации программ*. – М.: Радио и связь, 1988, – 256 с.

2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. *Приемы объектно-ориентированного проектирования. Паттерны проектирования.* – Питер, 2001. — 368 с.

3. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. *Лекции по теории графов.* – М.: Наука, 1990. 384 с.

4. Райгородский А.М. *Модели случайных графов.* – М.: МЦНМО, 2011. 136 с.

5. Шилов Н.В.; Конюхов И.В.; Холодов Я.А.; Ягудин З.Р.; Масягин С.В. *Лабораторные работы по математике для студентов-программистов*. — В сб.: Математическое образование в школе и вузе: опыт, проблемы, перспективы (MATHEDU'2021). Материалы X Международной научно-практической конференции (Казань, 22-28 марта 2021 г.) / отв. ред. Л.Р. Шакирова. - Казань: Издательство Казанского университета, 2021. — Стр.227-238.