УДК 372.862

ОСНОВНЫЕ ПРОБЛЕМЫ ИЗУЧЕНИЯ ПРОГРАММИРОВАНИЯ В СОВРЕМЕННОЙ ВЫСШЕЙ ШКОЛЕ

Фокин Р.Р.

ФГКВОУ ВО «Военно-космическая академия имени А.Ф. Можайского», Санкт-Петербург, e-mail: rrfokin@yandex.ru

Настоящая статья обобщает многолетний педагогический опыт работы в высшей школе автора статьи и его коллег. Статья адресована преподавателям информатики, информационных технологий, математики, в особенности преподавателям программирования и вычислительной математики, а также студентам, аспирантам, соискателям ученых степеней, для которых актуальны или будут актуальны методики преподавания в высшей школе указанных выше областей знания. В статье предлагается, по мнению ее автора, обзор важнейших проблем преподавания программирования в современной высшей школе, а также пути хотя бы частичного их решения. В качестве основных проблем рассматриваются проблемы, имеющие как психолого-педагогические, так и общественно-политические корни. Актуальность рассмотрения проблем преподавания программирования определяется как тем, что именно программирование (в отличие от прочих разделов информатики) является (вместе с математикой) полюсом недоступности для понимания большинства студентов, так и огромной роли программирования в современном обществе. Предлагаемые автором подходы не обеспечивают полного решения указанных проблем. В большинстве случаев лишь намечаются пути, где следует искать, по мнению автора, решение указанных проблем. В отдельных случаях предлагаются конкретные методики преподавания, эффективность которых находит подтверждение в конкретной практике преподавания программирования автором и его коллегами. Эти методики основаны на сервисном подходе к преподаванию, подходе, использующем авторские сервисы обучения программированию.

Ключевые слова: сервисы обучения, RAD-программирование, информатика, вычислительная математика, виды мышления, информационные технологии, пакеты прикладных программ

THE MAIN PROBLEMS OF LEARNING PROGRAMMING IN A MODERN HIGH SCHOOL

Fokin R.R.

Federal State Public Military Institution of Higher Education «Military Space Academy named after Mozhaisky», Saint-Petersburg, e-mail: rrfokin@yandex.ru

This article summarizes the many years of teaching experience in higher education of the author and his colleagues. The article is addressed to teachers of Informatics, information technologies, mathematics, especially teachers of programming and computational mathematics, as well as students, postgraduates, applicants for academic degrees, which are relevant or will be relevant methods of teaching in higher education of the above areas of knowledge. According to the author, the article offers an overview of the most important problems of teaching programming in modern higher education, as well as ways to at least partially solve them. Both, the problems with psycho-pedagogical roots and the problems with socio-political roots are considered as the main problems. The relevance of the problems of teaching programming is determined both by the fact that programming (unlike other branches of computer science) is (along with mathematics) the pole of inaccessibility to the understanding of most students, and the huge role of programming in modern society. The approaches proposed by the author do not provide a complete solution to these problems. In the most cases, the ways, where solutions to these problems should be sought, are only outlined, according to the author opinion. In some cases, specific teaching methods are proposed, the effectiveness of which is confirmed in the specific practice of teaching programming by the author and his colleagues. These methods are based on the service approach to teaching, an approach that uses the author's services of teaching programming.

Keywords: learning services, RAD-programming, computer science, computer mathematics, types of thinking, information technology, application packages

Информатика как новая область знаний (ОЗ) упоминается с 1970-х гг. и, в частности, включает в себя программирование. В эту ОЗ входят научные дисциплины, не похожие друг на друга по своей методологии, а следовательно, и по методике преподавания. Программирование по методологии и методике преподавания близко к математике [1–3], но есть в информатике и технические дисциплины, и гуманитарные. Методика преподавания информатики – это общее место. Средства массовой информации пишут о большой востребованности информатики и программирования в современном обществе. Что касается программирования – это не совсем так.

Цель исследования: выявить основные проблемы изучения программирования в современной высшей школе и указать пути их возможного решения. Эти проблемы слишком сложны, чтобы одна статья могла их все полностью решить.

Материалы и методы исследования

Основной метод нашего исследования – индукция (путь от частного к общему), вспомогательный –

дедукция (путь от общего к частному). Только точные науки могут использовать дедукцию в качестве основного метода. Следует заметить, что индукция не всегда приводит к истине, за исключением математической индукции, которая обычно не применима нигде, кроме математики. Поэтому наши выводы абсолютизировать нельзя. Материалами для индуктивных рассуждений являются приводимые ниже частные примеры. Выбор именно этих примеров имеет ключевую ценность для результатов исследования.

Результаты исследования и их обсуждение

Программист в настоящее время не является широко востребованным специалистом как в России, так и за рубежом. Отсюда проблемы при обучении программированию. Пример 1: экзамен по программированию. Диалог преподавателя (П) и студента (С). П: «Что такое инкапсуляция?» С: «Ну, понимаете, компьютер обрабатывает информацию, у него есть процессор для этого ...» П: «Стоп, инкапсуляция Вам не знакома!» С: «Зачем мне вообще программирование? Я диплом получу, устроюсь работать сисадмином!» Системный администратор (сисадмин) обычно не программирует, причем эта профессия сейчас наиболее востребована для специалистов в области информатики. Программирование – это малая часть современной информатики. Сегодня на кафедре информатики или информационных технологий (ИТ) из 20–30 преподавателей программировать умеют лишь 2–3 человека. Из выпускников приблизительно такая же доля работает программистами, остальные – сисадминами, ИТ-менеджерами и т.п. Среди web-разработчиков (их меньше, чем сисадминов, но тоже много) большая доля web-дизайнеров, web-программистов мало.

О социальной значимости [2] программирования. Пример 2: известен случай, произошедший в США в 1960-е гг. На старте взорвалась ракета с искусственным спутником Земли. Это произошло из-за ошибочного оператора управляющей программы на Фортране. Вместо Y = X + 5 было введено Y = X + S, буква S внешне похожа на цифру 5. Это семантическая, а не синтаксическая ошибка (ошибка смысла, а не нарушение формальных правил языка Фортран) – такие ошибки автоматически не обнаруживаются.

Пример 3: вот опыт, который может произвести каждый, кто пользуется пригородной электричкой. Пусть имеются 3 станции с номерами, расположенные по ходу электрички в следующем порядке: 1, 2, 3. Предположим, Вы купили билет от станции N_1 до станции N_2, причем $N_1 = 1$, $N_2 = 3$. Допустим, по некоторой причине Вы решили выйти на станции N = 2 (в середине дороги). Автоматический турникет Вас не выпустит. Но около автоматического турникета стоит сотрудник, который по Вашей просьбе проверяет Ваш билет и в ручном режиме Вас выпускает. Почему так происходит? На билете имеется штрих-код, это очень большое натуральное число, которое однозначно идентифицирует конкретный билет. В момент покупки билета на центральный компьютер (сервер) данной железной дороги делается запись о том, что данный билет был куплен в некоторые время и дату от станции $N_1 = 1$ до станции $N_2 = 3$. Вы выходите на станции N, автоматический турникет посылает сообщение на центральный сервер (на десятки километров) с просьбой проверить Ваш билет. Когда-то не очень дальновидный программист запланировал на сервере выполнение следующего оператора: Если $N_2 = N$ Тогда «Выпустить пассажира» Иначе «Пассажира не выпускать». В нашем случае при $N = 2$ и $N_2 = 3$ турникет получит от сервера (опять на десятки километров) команду «Пассажира не выпускать». Поэтому такие турникеты срабатывают с задержкой в 10–15 сек. В результате образуется очередь на выход. Некоторые полагают, что в каждый билет встроен микропроцессор, это не так, там только штрих-код. Следовало бы в соответствующей программе предусмотреть проверку более сложного условия, что N лежит между N_1 и N_2, которое математически выражается так $(N - N_1)(N_2 - N) \geq 0$. Прошло более 10 лет со времени появления таких турникетов, а ошибка так и не исправлена до сих пор.

Закончилось противостояние двух мировых систем – капитализма и социализма. Снизилась вероятность мировой войны. Снизилась интенсивность гонки вооружений. Следствие этого – снижение спроса на исследования математического, естественнонаучного, технического направления, повышение спроса на исследования маркетингового, экономического, психологического, социально-психологического, в целом – гуманитарного характера. Пример 4: мои коллеги, которые некоторое время работали программистами и математиками в США и в других странах, ныне почти все вернулись в Россию. По их мнению, например, в США снизилась потребность в высококвалифицированной рабочей силе и повысилась – в низкоквалифицированной. В связи с глобальной информатизацией повысилась потребность в программистах, которые создают программное обеспечение низкого качества в больших объемах и работают за относительно низкую зарплату дистанционно, не выезжая из своих стран. Исключение составляют специалисты низкоуровневого программирования – это про-

граммисты в кодах конкретного процессора или на ассемблере. Это машиннозависимое программирование, оно привязано к архитектуре конкретного процессора. В последние 10 лет появились многочисленные нетиповые архитектуры – это встроенные RISC-процессоры, управляющие технологическими процессами и гаджетами (планшетами, смартфонами, фотоаппаратами, принтерами и т.п.). В основном они программируются на языке высокого уровня Java – это оптимально с точки зрения себестоимости программного обеспечения. Но Java-машину при этом нужно привязывать к конкретной архитектуре, следовательно, без процессорных кодов и ассемблера не обойтись. Для этого требуются низкоуровневые программисты, которые в результате перманентного повышения своей квалификации становятся незаменимыми сотрудниками для своей компании.

В СССР выпускались многочисленные микропроцессорные наборы, которые управляли исполнительными механизмами в технологических процессах и периферийными устройствами (принтерами, сканерами). Соответственно, были нужны высококвалифицированные низкоуровневые программисты в большом количестве. В современной России около 5 лет назад начали производить микропроцессорный набор на базе архитектуры процессора советского суперкомпьютера Эльбрус-3. На базе современных нанотехнологий его сделали микропроцессором. Среди чипов массового производства по быстродействию он занимал на то время 2 место в мире, уступая лишь Intel Core Duo. Из открытой печати у автора иных сведений по этому вопросу нет. Нужны ли сейчас в России высококвалифицированные низкоуровневые программисты в значительном количестве? Сомнительно. Автор статьи является низкоуровневым программистом Советских микропроцессорных наборов К1810 (аналог Intel 8086) и К1801 (аналог DEC PDP-11), имеются соответствующие дипломы. На основе имеющегося опыта можно сказать, что научиться низкоуровневому программированию только по учебникам невозможно, нужно читать заводскую документацию.

Еще одним следствием того, о чем сказано выше, является снижение уровня образования в США и в других странах. На основе этого перейдем к конкретной проблеме преподавания программирования – оно требует наличия соответствующего аппаратного (hardware) и программного (software) обеспечения. В университетах, где автору приходилось работать, закупалось в достаточном количестве hardware и оно содержалось в удовлетворительном состоянии, чего нельзя сказать о software. Лет десять назад с торрентов можно было скачивать контрафактный software и использовать его на занятиях. Теперь это делать строго запрещено под угрозой уголовной ответственности для преподавателя. Необходимый software реально вузами не закупается. Как можно изучать современное программирование при отсутствии систем программирования?

Программирование и математика вызывают негативную реакцию [1–3] нескольких поколений студентов по крайней мере последние полвека. Информатики полвека назад не было, но было программирование, причем почти во всех вузах. Пример 5: выпускники 1981 г. Института культуры имени Н.К. Крупской по специальности «Библиограф» в дипломе имели дисциплину «Программирование», изучали язык «Фортран». В те годы считалось, что каждый специалист с высшим образованием должен уметь пользоваться компьютером для выполнения расчетов, связанных с его профессиональной деятельностью. Понятия пользователя и прикладного программиста были тождественными. В настоящее время существует понятие «парапрограммист» – программирующий пользователь. Им может быть даже гуманитарий, например врач, психолог, педагог. Новая научная теория, как правило, требует подтверждения экспериментом и статистической обработки его результатов. Нужны новые математическая, алгоритмическая модели и компьютерные программы. Поэтому неплохо ученому-гуманитарию также быть «парапрограммистом», «парастатистиком», «параматематиком». На практике никакого интереса среди современного студенчества к «парапрограммированию» не наблюдается. Они мотивированы получить специальность и высокооплачиваемую работу, а не разрабатывать новую научную теорию.

Согласно психологической теории [3–5] межполушарной асимметрии мозга (МАМ): 1. Существуют люди с преобладанием деятельности левого полушария мозга (мыслительный тип, левополушарные), правого полушария мозга (художественный тип, правополушарные), без явного преобладания деятельности какого-либо полушария (гармоничный тип). 2. Левое полушарие мозга отвечает за обработку вербальной информации (выраженной заранее оговоренными сигналами), правое – невербальной (непосредственной) информации. 3. Левополушарные склонны к точным, естественным наукам, к технике, несколько замедленно оценивают сложную ситуацию путем анализа и логических рассуждений. Право-

полушарные склонны к гуманитарным наукам, к высоким спортивным достижениям (кроме, например, шахмат), очень быстро схватывают сложную ситуацию в целом путем интуиции.

Пример 6: экзамен по высшей математике. Диалог преподавателя (П) и студента (С). П: «Множество М = {–1, 0, 5}, назовите какую-нибудь верхнюю грань М» С молчит. П: «Что такое верхняя грань?» С: «Верхняя грань числового множества – это любое число, большее либо равное любому из его элементов» – вызубрил! П: «Из чисел 10 и –10 какое является верхней гранью?» С после долгого раздумья: «Наверное, 10». П: «Правильно! Это точная верхняя грань или нет?» С: «Точная» П: «Что такое точная верхняя грань?» С: «Точная верхняя грань числового множества – это наименьшая из всех его верхних граней» – вызубрил! Студент учебник читал, материал запомнил. Что он еще мог сделать? Ничего! Такие диалоги характерны для более чем половины студентов всех профессиональных направлений. Студент из примера 6, вероятно, правополушарный [3], таких в настоящее время большинство – вот откуда широкое неприятие программирования и математики.

По мнению многих психологов [4, 5], среди всех людей доля гармоничных очень мала (≈2%), доли правополушарных и левополушарных приблизительно одинаковы (≈49%), Среди людей, связанных с точными, естественными, техническими науками, больше доля левополушарных, среди людей, связанных с гуманитарными науками, больше доля правополушарных. Это не соответствует [3] действительности: 1. В настоящее время правополушарных значительно больше даже среди студентов физико-математического и технического направлений. 2. Эти доли меняются во времени, последние 20 лет по всем профессиональным направлениям сильно растет доля правополушарных студентов, растет доля гармоничных, сильно падает доля левополушарных. Автор статьи располагает взятыми из интернета данными 1960-х гг. про один из классических университетов. Там в целом по вузу была доля левополушарных студентов больше половины. В то время «мыслители» были популярнее «художников»!

Согласно психологической теории Б.М. Теплова [6] о 4 типах мышления, мышление бывает: с одной стороны, конкретным (действенным) или абстрактным; с другой стороны, понятийным или образным. Отсюда 4 типа: действенное понятийное; действенное образное; абстрактное понятийное; абстрактное образное. Теория МАМ и теория Б.М. Теплова о 4 типах мышления интересны тем, что относятся к экспериментальной психологии, а не к концептуальной. Они подтверждены экспериментами. Педагогический принцип наглядности состоит в том, что при росте степени наглядности учебного материала результаты обучения улучшаются. Это не соответствует [1] действительности! С ростом наглядности результаты обучения сначала улучшаются, достигают некоторого максимума, а затем начинают убывать. Автором статьи проводились соответствующие эксперименты. Так что для всякого учебного материала существует оптимальная степень наглядности. Ее превышение вызывает ухудшение результатов обучения. Почему сначала происходит рост понимания с ростом наглядности – это объясняется во всех учебниках педагогики. Рост наглядности сначала немного разгружает два типа абстрактного мышления и немного нагружает два типа изначально разгруженного действенного мышления. Наглядность – это всегда демонстрация педагогом некоторого действия, которое происходит не в голове студента, а вне ее, например на экране компьютера. А если педагог все более увеличивает наглядность, пытаясь за счет этого все больше увеличить скорость подачи учебной информации? В некоторый момент все 4 вида мышления оказываются максимально загруженными, разгружаться им теперь некуда. Мозг быстро устает, степень понимания падает. Большая часть аудитории засыпает – это реакция здорового мозга на сильную усталость.

Улучшение качества обучения (программированию и математике в частности) путем увеличения степени наглядности учебного материала – естественный путь решения выявленных в статье проблем. При этом следует учитывать сказанное выше об оптимальной степени наглядности. В соответствии с теорией Б.М. Теплова наглядность может быть образной и понятийной. Для правополушарных студентов эффективнее будет работать образная наглядность, для левополушарных – понятийная. Понятийная наглядность обеспечивается подменой научной системы понятий учебной, что чревато негативными последствиями, в особенности если речь идет о высшей школе. Образная наглядность в этом смысле безопаснее, в настоящее время она также более актуальна, поскольку правополушарных студентов большинство, их доля растет во времени.

Заключение

Широко применяемые ныне методики преподавания программирования оптимизированы для обучения левополушарных студентов, они сложились фактически в 1960-е гг., когда таких студентов было боль-

шинство. Сейчас в большинстве правополушарные студенты, они нуждаются в иных методиках преподавания программирования, основанных на образной наглядности. Внедрение в обучение современных технологий программирования (визуально-ориентированного, событийно-ориентированного, сценарного программирования), внедрение наглядных информационно-компьютерных технологий обучения обеспечило бы образную наглядность.

Эти решения предлагается основывать на использовании сервисов обучения программированию. Сервис обучения – это новое понятие педагогики, разработанное и обоснованное к практическому применению автором статьи и его коллегами [1, 2] на базе многолетней практики преподавания информатики и математики начиная с 2002–2003 гг. Существенным подкреплением этих идей [1] послужило появление [7, 8] начиная приблизительно с 2004 г. зарубежных и отечественных научных работ, посвященных возникновению новой ОЗ – науки о сервисах, управлении и инжиниринге, по английски – Service Science, Management and Engineering (SSME).

Системы программирования [1, 2], поддерживающие указанные выше современные технологии, обычно очень дороги. Установка одной такой системы на один учебный компьютер стоит десятки или сотни тысяч рублей. Поэтому важно использовать программы содействия образованию некоторых крупных коммерческих компаний. Например, компании Microsoft и IBM полноценные версии почти всех своих программных продуктов предоставляют бесплатно вузам, студентам и преподавателям всех стран. С пользователя требуется лишь обязательство применять полученный бесплатно программный продукт только в образовательных и научных целях. Microsoft – владелец Visual Studio (многоязыковой системы программирования). IBM – владелец Matcad, Matlab, SPSS-Statistica (математических пакетов прикладных программ с возможностями программирования). Российская компания 1С бесплатно предоставляет всем желающим функционально сокращенные учебные версии всех своих программных средств. К сожалению, таких компаний, как Microsoft, IBM, 1С, немного.

Список литературы

1. Абиссова М.А., Фокин Р.Р. Сервисы обучения информатике и информационным технологиям в высшей школе. СПб.: Изд-во СПбГУСЭ, 2010. 195 с.

2. Абиссова М.А., Атоян А.А. Сервисы обучения RAD-программированию для активизации познавательной деятельности студентов при обучении информатике и математике // Письма в Эмиссия.Оффлайн. 2013. № 12. [Электронный ресурс]. URL: http:// www.emissia.org/ offline/ 2013/ 2118.htm (дата обращения: 16.09.2019).

3. Фокин Р.Р. Некоторые психологические и статистические аспекты преподавания дисциплин из областей математики и информатики в современной высшей школе // Современные наукоемкие технологии. 2019. № 9. С. 175–179.

4. Спрингер С., Дейч Г. Левый мозг. Правый мозг. М.: Книга по требованию, 2013. 254 с.

5. Москвина Н.В., Москвин В.А. Межполушарные асимметрии и индивидуальные различия человека. М.: Смысл, 2011. 368 с.

6. Маклаков А.Г. Общая психология. СПб.: Питер, 2016. 576 с.

7. Tadahiko A. What is Service Science? Japan Research Report. 2005, December. No. 246. [Electronic resource]. URL: https:// cyberleninka.ru/ article/ n/ nauka-ob-uslugah-voprosy-prepodavaniya-novoy-distsipliny (date of access: 16.09.2019).

8. Сорокин А.В. Новая наука о сервисах, управлении и инжиниринге // Экспертная сеть по вопросам государственного управления: электронное справочное пособие. 2010. [Электронный ресурс]. URL: http://www.gosbook.ru/ node/10083/ (дата обращения: 16.09.2019).