Мир науки. Педагогика и психология / World of Science. Pedagogy and psychology https://mir-nauki.com

2022, №6, Том 10 / 2022, No 6, Vol 10 https://mir-nauki.com/issue-6-2022.html

URL статьи: https://mir-nauki.com/PDF/33PDMN622.pdf

Ссылка для цитирования этой статьи:
Пирогов, В. Ю. Обучение программированию в высшей школе — проблемы и решения / В. Ю. Пирогов // Мир науки. Педагогика и психология. — 2022. — Т. 10. — № 6. — URL: https://mir-nauki.com/PDF/33PDMN622.pdf

For citation:
Pirogov V.Ju. Programming training in higher education — problems and solutions. *World of Science. Pedagogy and psychology*. 2022; 10(6): 33PDMN622. Available at: https://mir-nauki.com/PDF/33PDMN622.pdf. (In Russ., abstract in Eng.).

Пирогов Владислав Юрьевич
ФГБОУ ВО «Шадринский государственный педагогический университет», Шадринск Россия
Профессор кафедры «Программирования и автоматизации бизнес процессов»
Кандидат физико-математических наук, доцент
E-mail: Vladislav-133@yandex.ru

Обучение программированию в высшей школе — проблемы и решения

Аннотация. Вопросы преподавания программирования в высшей школе тесно связаны с возрастающей нехваткой ИТ-специалистов и программистов, в частности, в российской экономике. Амбициозные задачи, которые встали в области разработки программного обеспечения, требуют все больше специалистов. В статье рассматриваются трудности, с которыми встречается преподаватель программирования и инструменты, которые можно использовать для повышения эффективности процесса преподавания. Автор отделяет алгоритмическое мышление от технологии программирования. Показывается, что многие ошибки начинающих программировать связаны с непониманием того, как функционирует компьютер и каким образом происходит выполнение программы. Обращается внимание на то, что начинающий разработчик не всегда понимает разработанную им программу, отсюда возникают проблемы в такой важной компетенции любого программиста, как умение отлаживать программу. Указывается также на проблему резкого неравенства в способности программировать в студенческих группах. Также отмечается проблема обучения программированию, связанная с пониманием предметной области. Во второй части статьи рассматриваются возможности преподавателя в выборе способов улучшения качества обучения программированию. В частности, разбирается вопрос выбора языка программирования. Разбираются аргументы, которые используются при выборе языка и отмечается, что выбор языка программирования связан более с технологиями программирования, нежели с развитием алгоритмического мышления. Далее рассматривается среды программирования и указывается, что они мало влияют на сам процесс и результат обучения. Автор подробно останавливается на мотивации студентов, видах мотивов. Указываются способы повышения мотивации. Далее автор останавливается на взаимодействии «преподаватель — студент» и важности такого приема в обучении, как автоматизация проверки заданий, выполняемых студентами.

Ключевые слова: программирование; программист; языки программирования; отладка программы; тестирование программы; обучение программированию; среда программирования; обучение

Введение

Превышение спроса на ИТ-специалистов над предложением по мнение многих специалистов может продлится до 10 и более лет[1]. Особенно это касается специалистов в области разработки программного обеспечения. Проблема в значительной степени усугубляется, с одной стороны, принятыми против России санкциями, в том числе и по линии закупок программного обеспечения, а с другой курсом на широкомасштабное импортозамещение в области разработки программного обеспечения и электроники. Задачи поставлены масштабные и выполнение их требует в том числе наличия достаточного количества специалистов в области программирования и организации разработки программ. Для того, чтобы увеличить насыщение рынка специалистами правительством были разработаны ряд мер. В частности, было предложена программа бесплатного обучения программированию школьников[2]. И это к тому, что подобные программы уже есть в стране и у крупных вузов (МГУ, Инополис) и корпораций (Академия Яндекс Лицей, Al-академия от Сбер). Еще одно направление работы в области подготовки специалистов в области программирования — крупные предприятия, которые сами берутся за обучение по необходимым им дисциплинам. Наконец, в вузы по специальностям, связанным с информационными технологиями в последнее время значительно увеличено количество бюджетных мест.

Конечно, подготовка специалистов в области информационных технологий и программирования — это все важные инструменты, которые дают определенные результаты. Но надо иметь в виду, что обучение можно представить в виде черного ящика, где на входе те, кто будет обучаться, а на выходе уже обученные. И количество, и качество прошедших обучение не всегда может соответствовать поставленным требованиям. Чему и как учить в области программирования становится одной самых важных проблем подготовки современных специалистов в ИТ-области. Что касается специалистов в области программирования, то здесь особая задача: овладение технологиями не всегда является первостепенной проблемой обучения. Часто трудность заключается в том, что обучаемый плохо понимает логику работы компьютера (исполнителя) во время выполнения программы, и в этом случае написание собственных программ становится для него непреодолимым препятствием [15].

В рамках данной статьи мы рассмотрим некоторые трудности обучения программированию в Высшей школе [10]. Особенности восприятия обучаемыми процесса программирования, процесса выполнения программы, а также процессов отладки и тестирования программного обеспечения. Мы рассмотрим возможные подходы, позволяющие повысить эффективность обучения программированию.

Трудности обучения программированию

Обучение программированию всегда происходит по двум направлениям: алгоритмическое мышление и технология. Под технологией понимается целый набор инструментов: язык программирования, среда разработки, средства отладки и анализа кода, библиотеки и взаимодействие с операционной системой. Под алгоритмическим мышлением будем понимать умение строить последовательность действий, выполнение которых приведет к поставленной цели (см. также [8; 9]). Конечно, человек в течении своей жизни постоянно строит такие последовательности для достижения целей. Суть, однако, в том, что обычный

[1] По утверждению Росстата России может дополнительно потребоваться до одного миллиона ИТ-специалистов: https://rg.ru/2022/04/27/rosstat-rossii-dopolnitelno-trebuetsia-bolshe-milliona-it-specialistov.html.

[2] По замыслу правительства школьники 8–11 классов будут обучаться программированию за счет государства http://government.ru/news/45921/.

человек строит жизненные алгоритмы руководствуясь не точной логикой, интуицией, эмоциональным выбором и т. п. Обучаемый программированию сталкивается с тем, что ему необходимо изменить его способ создания алгоритмов. Во-первых, в данном случае алгоритм должен однозначно приводить к достижению поставленной цели. Во-вторых, алгоритм выполняет компьютер, а не человек. Профессиональному программисты эти проблемы неведомы, но для обучаемого, они могут стать серьезным препятствием.

Важно увидеть границу между алгоритмом и языком программирования [16]. Приведем два примера.

Пример 1. Присвоение значения переменной — это элемент алгоритма. Но в разных языках такое присвоение может осуществляться по-разному, в том числе и несколькими способами. Например, в языке C, и a = 10; и a = b = 10; и int a = 10; дают один и тот же результат, но это все является присвоением значения переменной.

Пример 2. Поиск простого элемента в массиве в языке программирования может быть реализован разными способами: различные циклы, функции поиска, методы объекта «массив». Для профессионального программиста раскрытие этого элемента происходит только на стадии написания программы. Данный пример интересен тем, что по мере развития навыков программирования определенные алгоритмы для обучаемого начинают «сворачиваться», становиться простым элементом, который «разворачивается» только при написании программы. Для начинающего же программировать, сам алгоритм поиску может представлять определенную трудность, не говоря уже о возможности записать его в рамках данного языка программирования разными способами.

В работе [11] выделяется три класса типичных ошибок обучаемых программированию, связанных непосредственно с пониманием того, как работает компьютер.

Ошибки параллелизма. Обучаемый предполагает, что компьютер воспринимает программу в целом, вне зависимости от того, какая в данной момент строка программы выполняется. Т. е. компьютер способен сам при необходимости обратиться к нужной строке программы, которая уже выполнена, или же одновременно выполнять несколько строк.

Ошибки интенциональности (направленности). Обучаемый ошибочно делает выводы о будущих действиях компьютера на основе не всей программы, а только одного его фрагмента. По этой причине обучаемый не верно трактует смысл остальной программы.

Ошибки эгоцентризма. Обучаемый ошибочно считает, что компьютер способен сам каким-то образом «понимать» цели данной программы. Обычно это выражается в том, что студент пропускает некоторые важные элементы (условия, циклы), считая, что компьютер сам воспроизведет эти элементы.

Программирование представляет собой единство нескольких видов деятельности: написание программы, тестирования и отладки. Тестирование, как процесс проверки соответствия программы набору определенных требований, довольно часто в Вузах сводят к автоматической проверке программы по заранее определенным тестам. Этот подход, который стал интенсивно использоваться в последнее время, несомненно, является важным методом, повышающим качество образования в области программирования. Но при этом упускается одно важное звено в обучении: студент не участвует в разработке тестового материала для задачи. При этом, если обучаемый имеет доступ к набору тестов, то возникает разрыв между условием задачи и разрабатываемой им программой. Мы вернемся к вопросу о тестировании разрабатываемых студентами программ в следующей части нашей статьи.

33PDMN622
Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

Упомянутая выше отладка программы, тесно связана с пониманием программного текста. Основная задача отладки — локализовать имеющиеся в программе ошибки. Очень часто начинающие программисты решают поставленную перед ними задачу, связанную непосредственно с программированием, но оказываются абсолютно беспомощны перед проблемой поиска и понимания ошибок. Некоторые авторы утверждают [7], что и опытные программисты часто обладают определенными пробелами в области отладки. При этом те, кто успешен в отладке, обычно хорошо программируют. Этот феномен можно объяснить только тем, что написание программы и понимание текста программы не всегда идут параллельно друг другу. Почему это происходит? Можно указать, как минимум, четыре причины:

1. При написании программы программист часто использует уже готовые шаблоны. При чем не важно вставляет ли он в свою программу уже готовый фрагмент, или этот шаблон, по сути, берется из его памяти.
2. Это может показаться удивительным, но очень часто программист при написании пользуется интуицией, основанной на уже имеющемся у него опыте в области программирования. Результат такого подхода, по сути, сводится к результату из пункта 1.
3. Если программа достаточно велика, то смыслы отдельных строк, фрагментов, структур могут через некоторое стать для программиста не совсем понятны, поскольку они были встроены в некую общую логику, которая создавалась в процессе написания программы. Отдельные составляющие этой логики через некоторое время оказываются утерянными.
4. Языки программирования (их реализации) обладают большим количеством особенностей, которые можно почерпнуть только из хорошего справочного руководства[3]. Никто из программистов (и тем более обучаемых) не сверяет каждый элемент (оператор, функцию), который он использует, со справочником. Ошибка как раз и может возникнуть из-за того, что была упущена некоторая особенность языка программирования.

Еще одной важной проблемой, которая возникает в преподавании программирования в Высшей школе неравномерность подготовки студентов. В последнее время многие школьники в рамках дополнительного образования получают очень неплохую подготовку в области программирования. Автору этих строк приходилось, в частности, встречаться со студентами, которые в свое время окончили курс Лицея Академии Яндекса. Эти студенты, будучи далеко не самыми лучшими учениками в лицее, в значительной степени превосходили своих одногруппников в Вузе по предметам программирования. Это интересное явление, встречается все чаще в наших высших учебных заведения и требует, видимо, какого-то нового подхода.

Важная проблема поднимается в статье [7]. Авторы, в частности, задаются вопросом, с чем связана способность человека вообще усвоить такую дисциплину, как программирования. В частности, говорится о слабой связи с общим образованием и уровнем интеллектуального развития. Авторы считают, что предрасположенность (или непредрасположенность) к такой деятельности как программирование заложено у некоторых изначально. Это спорное утверждение, однако, не снимает такой вопрос, как профориентация. В частности, дополнительное школьное образование, проектная деятельность в школах, система олимпиад и могут осуществить предварительный отбор школьников для будущей деятельности в области разработки программного обеспечения.

[3] Довольно часто программист изучения справочника заменяет собственным не большим исследованием.

Есть группа проблем, связанных с освоением конкретных элементов программирования. Например, переменные, присвоение, условные конструкции, операторы выбора, циклы, сложные структуры. Для более точного анализа мы предлагаем разбить трудности в программировании на следующие классы:

1. Простые переменные (типы) и присвоения. Константы. Выражения.
2. Условные конструкции, операторы выбора, циклы[4].
3. Сложные структуры. Речь идет и о структурах, которые есть в конкретном языке программирования и универсальных структурах: списки, стек, очередь и т. д.
4. Функции, параметры функций, локальные и глобальные переменные, вложенность функций.
5. Рекурсивные алгоритмы.
6. Параллельное программирование.
7. Программирование файловой системы.
8. Библиотеки и разработка библиотек. Статические и динамические библиотеки.

Следует отметить, что мы в перечисленных пунктах отметили только разделы, имеющие отношение к фундаменту программирования. Есть большие и сложные разделы в программировании, такие как оконные приложения, сетевое программирование, web-программирование, программирование мобильных устройств, программирование микроконтроллеров. Лишний раз подчеркнем, однако, что эти разделы нельзя освоить, если не будут освоены фундаментальные разделы, которые были перечислены выше. Следует отметить, что пункты 5–6 относятся к так называемым сложным вопросам программирования. Проблемы здесь заключается как раз в алгоритмическом мышлении и понимании того, как работает программа. Пункты 7–8 это типичные технологические вопросы программирования.

Часто проблема написания программы у начинающих программистов корениться не в самой программе, ее логике, языке программирования, а в понимании условия задачи. Другими словами, студент не всегда понимают ту предметную область, с которой имеет дело, решая задачу. Часто проблема у современного студента коренится в умении читать и понимать тексты.

Возможные решения

Рассмотрим различные подходы, которые можно использовать для повышения эффективности обучения программированию.

Языки программирования

Влияет ли выбор языка на освоение основ программирования? Вопрос это обсуждается очень часто. Но при ближайшем рассмотрении выясняется, что выбор языка программирования увязывают не столько с умением разрабатывать и записывать алгоритмы, сколько с другими аспектами. Перечислим их.

[4] На наш взгляд при преподавании следует совмещать условные конструкции и циклы, так как в основе тех и других лежит принцип изменения последовательности выполнения команд.

33PDMN622
Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

1. Есть простые и сложные в усвоении языки программирования. Для начинающих программировать следует брать язык простой в изучении. Довольно часто в последнее время, особенно это касается школьного образования, в качестве такого языка выбирают Python. На самом деле есть более простые языки программирования, созданные специально для обучения. Но всегда возникает вопрос: разве учиться программированию сводится к только к изучению языка. Алгоритм написания сортировки (любой) для всех языков будет иметь приблизительно одну и ту же сложность в понимании обучаемого. Иногда в качестве примера приводят программу, выводящих строку «Hello world!» на разных языках программирования [12; 13]. Наиболее простая (короткая) программа и приводится в качестве индикатора того, что язык более прост и следует начинать именно с него. В действительности это ошибочный аргумент, поскольку для любого языка можно взять простейшую программу и принять ее в качестве шаблона, который потом можно использовать как основу для написания любой программы.

2. Еще один аргумент, который используется при аргументации выбора языка программирования связан со строгостью правил языка. В частности, речь идет о строгой типизации переменных. Среди части специалистов в области программирования бытует мнение, что первым языком должен быть язык, который бы был одновременно прост и подчиняющийся строгим правилам, как например строгой типизации или строгой структуре программе. К таким языкам, в частности, относится Pascal [5], который еще недавно был очень популярен в школьных курсах информатики[5]. Логика сторонников такого подхода заключается в том, что обучаемому с самого начала будет привит набор правил, привычка к соблюдению которых будет сказываться в дальнейшем на его профессиональную деятельность. Но и эта аргументация не имеет прямого отношения к базовым умениям любого профессионального программиста — умению создавать и записывать алгоритмы.

3. Данный довод также иногда приходится встречать, особенно в Интернет-статьях. Речь идет о том, чтобы не использовать специальные учебные языки, а выбирать сразу промышленный язык программирования, т. е. тот язык, на котором создается программное обеспечение. И в этом случае как раз на язык Pascal смотрят несколько настороженно, поскольку по мнению многих (хотя в общем это ошибочное мнение), он уже давно не используется в реальных разработках. Отчасти в этой есть своя логика, но трудно понять, почему «учебный» язык программирования может помешать дальнейшему усвоению других языков. Ну и вопрос об алгоритмическом мышлении по-прежнему остается не решенным.

4. Данный аргумент можно видеть в различных интернет-статьях о программировании [6] . В этих рассуждениях в основу кладется популярность языков программирования среди работодателей. Существуют специальные сайты, занимающиеся сбором информации о популярности языков программирования. Конечно, эта информация нужна не только тем, кто ищет работу, но и нам, работникам высшей школы, чтобы оперативно ориентироваться в быстро меняющемся ИТ-рынке. Но человек со слабым алгоритмическим мышлением не подойдет в качестве разработчика вне зависимости на каком языке он пытался писать программы.

[5] Следует отметить, что использование тех или иных языков программирования в обучении менялся практически каждое десятилетие. Был период, когда для этой цели использовался Basic [6], который сейчас, считается языком «второго» сорта.

[6] Можно привести одну из статей, посвященной выбору языка программирования: https://habr.com/ru/company/oleg-bunin/blog/488200/.

33PDMN622
Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

Заканчивая вопрос о выборе языка программирования для обучения, сошлемся на мнения руководителей групп разработчиков программного обеспечения [7], суть которого сводится к тому, что знание конкретного языка программирования все-таки не является главной компетенцией в профессиональной подготовке разработчика. Как правило такого же мнения придерживаются и многие профессиональные программисты. И так, язык программирования вряд ли облегчит задачу формирования у студентов алгоритмическое мышление, но может сформировать у обучаемого интерес и определенные навыки, имеющие отношение к написанию профессиональных программ.

Среда программирования

Насколько важна среда программирования (IDE)[8] при обучении? И здесь, как в случае с языками программирования часто выдвигается аргумент о том, что нужна наиболее развитая среда, с возможностью умных подсказок, запуска программы прямо из редактора и всеми отладочными инструментами. Выскажем ряд соображений.

1. Среда, которая снимает с обучаемого часть работы по написанию элементов языка, подсказывая возможные варианты, в-первых, частично создает пробел в профессиональных навыках; во-вторых, снимает с обучаемого ответственность за написанное, что может в реальности усложнить процесс отладки программы.
2. Среда разработки создает определенный барьер между программистом и средой исполнения. Ведь реальной средой исполнения является операционная система[9]. Ниже мы вернемся к этому тезису, когда будем говорить о мотивации студента.

На наш взгляд работа в средах программирования типа Eclips, PyCharm, NetBeans и др. интересна с точки зрения знакомства будущих специалистов с профессиональными инструментами разработки, с которыми в будущем, возможно, придется работать. На этом польза от выбора среды разработки, в сущности, и заканчивается.

Поскольку мы говорим о базовом обучении программированию, мы не останавливаемся на таком известном инструменте, как визуальное программирование. Данный инструмент, несомненно, хорош для создания оконных приложений [17], но, в сущности, базируется на базовых алгоритмических навыках программирования, а также такой важной парадигме, как объектно-ориентированное программирование.

Мотивация

Мотивация всегда называлась самой важной составляющей в обучении программированию. Рассмотрим несколько ее важных составляющих компонента (мотива) применительно к обучению программированию.

1. Наверное, самый важный мотив в учебе программированию является желание стать профессиональным программистом. Программист (разработчик программного обеспечения) в нашей стране довольно неплохо материально обеспечен и, как минимум, всегда может найти работу. А если учесть те процессы цифровизации, которые разворачиваются в России, то перспективы будущих программистов значительно повышаются. Кроме того, есть

[7] Мнения руководителей отделов программирования различных корпораций https://tproger.ru/experts/12/.

[8] Обычный термин: интегрированная среда разработки или integrated development environment (IDE).

[9] С web-приложениями дело обстоит несколько сложнее, но мы в данной статье имеем в виду обычные приложения, исполняемые в среде операционной системы.

перспективы поработать и в международных корпорациях, что для части студентов важный фактор. Автор этих строк довольно долго занимался мероприятиями, мотивирующих студентов к будущей профессиональной деятельности. К таким мероприятиям можно отнести: встречи с выпускниками вуза, работающими в области программирования, встречи с будущими работодателям, организация производственных практик на различных предприятиях в ИТ-отделах, организация работы по выпускным квалификационным работам, информирование студентов о востребованности специалистов в области информационных технологий в стране и в регионе. При этом часть ребят всегда мотивирована изначально, с первого курса и с ними никакой работы не требуется.

2. Выделим также мотив, который, я бы отделил от мотива, указанного выше. Принадлежность к сообществу профессиональных программистов, как к определенному слою общества, непосредственно не связанному с уровнем зарплаты, является важным фактором. С таким мотивом автору статьи приходилось встречаться не раз в 1990-е годы, когда работа по специальности далеко не всегда приносила программисту достаточный для безбедного существования доход. Способы стимулирования данного мотива такие же, как указывается в пункте 1. Но есть и еще один: это отношение преподавателей к ремеслу программиста.

3. Этот тип мотивации встречается довольно часто, но не является преобладающим. Можно назвать его: удовольствие от процесса программирования. Мы рассматриваем в статье только деятельность, связанную с процессом программирования, но в действительности этот мотив связан с более общим мотивом: удовольствие от интеллектуальной деятельности. Одним из способов развития подобной мотивации является выполнения заданий, связанных с элементами исследования. Например, студент должен написать программу, когда он заведомо не знает некоторые технологии, которые ему придется использовать. Элементами исследовательской работы, конечно, является олимпиадное программирование [2–4][10]. Еще одним важным блоком заданий, обладающих исследовательским потенциалом, конечно, является отладка программного обеспечения. Поиск и понимание ошибки, несомненно, является исследовательской работой. Особенно следует отметить задания, когда требуется выполнять отладку чужих программ.

4. Данный мотив следует отделять от мотива с номером 3. Этот мотив автор назвал бы «ожиданием нового». Он возникает всегда у части студентов, когда они начинают изучать новый язык программирования, новую технологию или парадигму. И в этой связи, как нам кажется, не следует делать упор на один язык программирования или одну парадигму. Необходима золотая середина. Если базовым языком программирования является, например, Pascal (Free Pascal), то не стоит стараться «выжать» из него, все возможные результаты: графику, сетевое программирование и т. п. Вполне можно вынести отдельные темы на курсовые проекты.

На наш взгляд есть еще один важный элемент, позволяющий усилить мотивацию студента к программированию. Это повышение значимости тех заданий, которые они выполняют. Эти задания часто довольно просты. Но суть в том, что обычно студент воспринимает выполнение заданий, просто как решение задач, а не создание программного обеспечения. И вот здесь таится возможность, имеющая отношение к мотивам с номерами 2 и 3. Сделать выполнение даже очень простого задания, как создание не большой и полезной утилиты. А для этого нужно как минимум предлагать студенту запускать программу в среде операционной системы, а не в интегрированной среде разработки. Приходилось встречать студентов, уже после определенного курса программирования, которые не знали, как запустить программу в операционной системе, ни вообще, где она у них в файловой системе лежит. Ясно,

[10] Мнения специалистов по разработке программного обеспечения об олимпиадном программировании однозначно положительно https://tproger.ru/experts/15/.

что для такого студента программирование является не процесс создания программного обеспечения, а только лишь выполнение заданий преподавателя. Автор многократно практиковал такой подход для начинающих программировать студентов:

1. Программа пишется в простом текстовом редакторе.
2. Компилируется или интерпретируется непосредственно средствами команд операционной системы.
3. Проверка написанных программ осуществляется непосредственно в операционной системе (обычный терминал или терминал в Meadnight Commander). Входные данные программа получает или на входном потоке, или в качестве параметров командной строки.

Студенты мгновенно схватывали всю суть данной технологии не зависимо от их начальной подготовки. К тому же понимание структуры файловой системы совсем не лишне для будущих профессионалов в области информационных технологий.

Парадигмы программирования

Касаясь проблемы обучения программированию, нельзя обойти такой важный вопрос как парадигмы программирования. При преподавании программирования, не обязательно исходить из исторических этапов развития программирования. При обучении следует выделить три основные парадигмы: процедурное программирование, модульное программирование, объектно-ориентированное программирование. На другие парадигмы: функциональное программирование, декларативное программирование, структурное программирование и др. — не обязательное уделять особое внимание в процессе преподавания[11]. Единственно, что здесь, на наш взгляд, важно с точки зрения увеличения эффективности преподавания — это последовательность изложения материала. Естественно, начинать с объектно-ориентированного подхода было бы не правильным решением. Представленная выше последовательность как раз является наиболее оптимальной, поскольку указанные парадигмы входя в следующую как подмножества.

Преподаватель и студент

Преподаватель и студент — это такая ключевая конструкция во всей системе высшего образования. Взаимодействие в этой «паре» меняется очень быстро за последнее время. Но мы не будем касаться вообще образовательного процесса. Отметим только то, что имеет отношение непосредственно к преподаванию программирования.

В области программирования особенно заметен следующее интересное явление. Некоторые студенты в группах значительно отрываются от общей массы даже если большая часть группы успешно усваивают материалы дисциплины. Как реагировать на такое явление: переводить особо успешных обучаемых на индивидуальные программы, использовать их в качестве помощников преподавателя или какой-то другой вариант? Мы не знаем ответа на этот вопрос. Однако это явление существует и оно, несомненно, влияет на весь процесс преподавания.

[11] Конечно, язык управления базами данных SQL является примером декларативного языка, но обычно он рассматривается в рамках других дисциплин: Базы данных, Информационные системы и т. п. [1].

Одним из важных нововведений, которое уже используется во многих Вузах — переход в части решения задач по программированию к автоматической проверке. Существуют общедоступные сервисы такие, например, как Yandex.Contest. Но любой из вузов может создать и собственный центр тестирования в области программирования. Для этого существует соответствующее программное обеспечение. Конечно, не любая программа подходит для автоматической проверки. Но программы для отработки базовых навыков программирования, в том числе и его важной части алгоритмического мышления, есть смысл проверять именно так. Для этого сами задачи должны быть сформулированы определенным образом, по принципу: входной — выходной поток информации. Что может дать такой подход?

1. Освободит преподавателя от необходимости нелегкой проверки программ студентов для того, чтобы уделять больше внимания индивидуальной работе, подготовке учебных материалов и т. д.
2. Научить студентов писать программы, которые точно соответствуют поставленным условиям.
3. Разорвать цепочку зависимости друг от друга преподавателя и студента в части проверки большинства заданий.

Предметная область

Понимание предметной области является необходимым условие корректного решения задачи. При обучении программированию предметная область определяется условием задачи. Часто условие формулируется кратко и четко. Например, дано положительное целое число. Необходимо найти все простые числа не превышающие данное число. Понятно, что для решения такой задачи необходимо знать определение простых чисел. Во всем остальном условие простое и ясное. Но если условие задачи достаточно велико, содержит определенную фабулу, нюансы задачи разбросаны по тексту условия, то многих студентов это ставит в тупик. Конечно, здесь проблема в умении читать и понимать тексты. Результат же такого неумения приводит к неполному пониманию предметной области. На наш взгляд для тренировки умения понимать текст и составлять полную картину предметной области, по крайней мере часть задач, должны иметь развернутое условие и желательно фабулу.

Выводы

Мы рассмотрели ряд проблем, связанных с преподаванием программирования в высшей школе:

1. Развитие алгоритмического мышления.
2. Не понимание работы компьютера.
3. Разрыв между процессом программирования и пониманием работы программы.
4. Неравномерность подготовки студентов.
5. Понимание предметной области.

Были выделены элементы преподавания, усовершенствование которых может улучшить качество преподавания:

1. Выбор языка программирования.
2. Выбор среды разработки программ.

33PDMN622
Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

3. Типы мотивации и способы воздействия на них.
4. Совершенствование взаимодействия между преподавателем и студентом.
5. Работа с текстами заданий и понимание предметной области.

В статье не затрагивались конкретные методики, которые могли бы увеличить эффективность преподавания конкретных вопросов программирования [14]. Автор надеется обратиться к ним в следующих статьях, посвященных преподаванию программирования в высшей школе.

ЛИТЕРАТУРА

1. Пирогов, В.Ю. Некоторые особенности преподавания языка управления базами данных / В.Ю. Пирогов. — Текст: непосредственный // Мир науки. Педагогика и психология. — 2018. — Т. 6. — № 6. — URL: https://mir-nauki.com/PDF/67PDMN618.pdf (accessed: 30.11.2022).
2. Пирогов, В.Ю. Некоторые вопросы преподавания программирования / В.Ю. Пирогов. — Текст: непосредственный // Мир науки. Педагогика и психология. — 2019. — Т. 7. — № 6. — URL: https://mir-nauki.com/PDF/34PDMN619.pdf (accessed: 30.11.2022).
3. Пирогов, В.Ю. Основы методики обучения рекурсивному программированию / В.Ю. Пирогов, И.В. Баландина. — Текст: непосредственный // Интернет-журнал «Мир науки». — 2018. — Т. 6. — № 4. — URL: https://mir-nauki.com/PDF/45PDMN418.pdf (accessed: 30.11.2022).
4. Пирогов, В.Ю. О возможностях дистанционного преподавания программирования для студентов технических специальностей / В.Ю. Пирогов. — Текст: непосредственный // Мир науки. Педагогика и психология. — 2021. — Т. 9. — № 6. — URL: https://mir-nauki.com/PDF/27PDMN621.pdf (accessed: 30.11.2022).
5. Алексеев, Е.Р. Free Pascal и Lazarus. Учебник по программированию / Е.Р. Алексеев. — Донецк.: УНИТЕХ, 2011. — 438 с. — Текст: непосредственный.
6. МакГрат, М. Visual Basic — программирование для начинающих / М. МакГрат. — Москва: Эксмо, 2017. — 192 с. — Текст: непосредственный.
7. Saeed Dehnadi and Richard Bornat. The camel has two humps (working title) — Text: electronic // School of Computing, Middlesex University. — UK. — 2006. — February 22. — URL: https://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf (accessed: 30.11.2022).
8. Парченкова, В.В. Алгоритмическое мышление и способы его развития / В.В. Парченкова // Молодежь и XXI век: материалы VIII междунар. молодеж. науч. конф. — Курск. — 2018. — С. 191–192. — Текст: непосредственный.
9. Босова, Л.Л. Как учат программированию в XXI веке: отечественный и зарубежный опыт обучения программированию в школе. / Л.Л. Босова — Текст: непосредственный // Информатика в школе. — 2018. — № 6(139). — С. 3–11.

10. Gomes, Anabela, Mendes, A.J. Learning to program — difficulties and solutions / Anabela1 Gomes, A.J. Mendes. — Text: electronic // International Conference on Engineering Education — ICEE. — 2007. — URL: http://icee2007.dei.uc.pt/proceedings/papers/411.pdf (accessed: 30.11.2022).

11. Konecki, M. Problems in programming education and means of their improvement / M. Konecki.: Daaam international scientific book, 2014. — Chapter 37. — pp. 459–470. — URL: https://daaam.info/Downloads/Pdfs/science_books_pdfs/2014/Sc_Book_2014-037.pdf (accessed: 30.11.2022) — Text: electronic.

12. Паволоцкий, А.В. Проблема выбора языка для начала обучения программированию в техническом вузе / А.В. Паволоцкий, Д.А. Королев, Н.И. Левицкая. — Текст: непосредственный // Качество Инновации Образование. — 2015. — № 12(127). — URL: https://publications.hse.ru/pubs/share/folder/r53cdc9c79/174489097.pdf (accessed: 30.11.2022).

13. Бобров, А.Н. Проблемы выбора языка программирования в школьном курсе информатики / А.Н. Бобров. — Текст: непосредственный // Молодой ученый. — 2015. — № 24(104). — С. 61–64.

14. Иванова, Л.В. Методы и формы обучения программированию в вузе / Л.В. Иванова, В.Е. Чекушина. — Текст: непосредственный // Сборник научных трудов SWorld. — 2013. — Т. 17. — № 3. — С. 18–22.

15. Кулигина, Н.О. Методика преподавания и оценки знаний студентов по программированию на кафедре АИС ДПИ НГТУ / Н.О. Кулигина. — Текст: непосредственный // Сборник трудов конференции Нижегородского государственного технического университета им. Р.Е. Алексеева. — 2014. — С. 198–202.

16. Абельсон, Х. Структура и интерпретация компьютерных программ / Х. Абельсон, Дж. Сассман, Дж Сассман. — Москва: КДУ, 2012. — 608 с.

17. Mehmet Fatih YIĞIT, Mustafa BASER. Learning Difficulties and Use of Visual Technologies in Learning to Program / YIĞIT Mehmet Fatih, BASER Mustafa // Participatory Educational Research (PER) Special Issue 2015-II, 2015. — 5–7 November. — pp., 27–3. — URL: https://www.perjournal.com/archieve/spi_15_2/4_per_15_spi_2_4_Page_27_34.pdf (accessed: 30.11.2022).

Pirogov Vladislav Jurievich
Shadrinsk Pedagogical University, Shadrinsk, Russia
E-mail: Vladislav-133@yandex.ru

Programming training in higher education — problems and solutions

Abstract. The issues of teaching programming in higher education are closely related to the growing shortage of IT specialists and, in particular, programmers in the Russian economy. The ambitious tasks that have arisen in the field of software development require more and more specialists. The article discusses the difficulties encountered by a programming teacher and the tools that can be used to improve the effectiveness of the teaching process. The author separates algorithmic thinking from programming technology. It is shown that many mistakes of beginners in programming are associated with a lack of understanding of how the computer functions and how the program is executed. Attention is drawn to the fact that a novice developer does not always understand the program developed by him, hence problems arise in such an important competence of any programmer as the ability to debug a program. The problem of a sharp disparity in the ability to program in student groups is also pointed out. The problem of teaching programming related to the understanding of the subject area is also noted. The second part of the article discusses the possibilities of the teacher in choosing ways to improve the quality of teaching programming. In particular, the question of choosing a programming language is examined. The arguments that are used when choosing a language are analyzed and it is noted that the choice of a programming language is more related to programming technologies than to the development of algorithmic thinking. Next, programming environments are considered and it is indicated that they have little effect on the process itself and the learning outcome. The author dwells in detail on the motivation of students, the types of motives. The ways of increasing motivation are indicated. Further, the author dwells on the interaction of "teacher — student" and the importance of such a technique in teaching as automation of checking tasks performed by students.

Keywords: programming; programmer; programming languages; program debugging; program testing; programming teaching; programming environment; teaching