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

2019, №6, Том 7 / 2019, No 6, Vol 7 https://mir-nauki.com/issue-6-2019.html

URL статьи: https://mir-nauki.com/PDF/34PDMN619.pdf

Ссылка для цитирования этой статьи:
Пирогов В.Ю., Попова Е.И. Некоторые вопросы преподавания программирования // Мир науки. Педагогика и психология, 2019 №6, https://mir-nauki.com/PDF/34PDMN619.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

For citation:
Pirogov V.Ju., Popova E.I. (2019). Some aspects of teaching programming. *World of Science. Pedagogy and psychology*, [online] 6(7). Available at: https://mir-nauki.com/PDF/34PDMN619.pdf (in Russian)

УДК 004.5

ГРНТИ 14.25.09

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

Попова Екатерина Игоревна
ФГБОУ ВО «Шадринский государственный педагогический университет», Шадринск, Россия
Доцент кафедры «Программирования и автоматизации бизнес процессов»
Кандидат экономических наук
E-mail: 978073@mail.ru

Некоторые вопросы преподавания программирования

Аннотация. Статья посвящена вопросам преподавания программирования в школе. Авторы высказывают предположение, что следует выделить обучение программированию в отдельную дисциплину. В статье рассматриваются алгоритмический и технологический аспекты программирования и баланс между ними в преподавании. Обсуждаются такие проблемы преподавания программирования, как содержание курса, возраст, с какого следует начинать преподавать программирование, подготовка учителей. Авторы подробно останавливаются на таком вопросе, как подбор используемого инструментария: язык, среда разработки, автоматизация проверки заданий. Предлагается, для обучения программированию использовать один из интерпретируемых алгоритмических языков программирования, например язык Питон. Указывается на определенные преимущества языка Питон в отношении к компилируемым языкам, таким как Паскаль. Особо отмечается необходимость автоматизации проверки программ, с помощью одного из специализированных сервисов для автоматического тестирования подобных заданий. В значительной степени это позволяет стандартизовать проверку программ и освобождает учителя для дополнительной работы с классом. Рассматривается вопрос о содержании курса программирования с точки зрения баланса между технологической и алгоритмической частями. Предлагается при изучении технологической части включать изучение алгоритмов, позволяющих заменить те или иные технологические возможности языка (сортировку, поиск и т. п.). Авторы подробно рассматривают одну из тем курса программирования – списки. Предлагается пример содержимого данной темы и фрагмент методической разработки авторов, одной из подтем (введение в списки). В представленном фрагменте подробно описывается последовательность введения понятия и соотношение между технологической и алгоритмической частями программирования.

Ключевые слова: программирование; преподавание программирования; языки программирования; язык программирования Питон; язык программирования Паскаль; алгоритмы; технология программирования

Введение

Вопрос подготовки специалистов в области программирования имеет мировой и национальный аспекты. О ближайшей нехватке специалистов в области программирования заговорили, например в таком IT-гиганте, как Microsoft[1]. Есть данные о нехватке специалистов в области программирования в знаменитой Кремниевой долине[2]. Но в России есть и свои особенности дефицита IT-специалистов. С 2019 года в нашей стране начинают действовать Национальные проекты, предназначенные для ускорения экономического развития и улучшения жизненного уровня российских граждан. Заметим, прежде всего, что исполнение любого из национальных проектов не возможно без подключения в том или ином виде IT-технологий. Следовательно, вложение денег в любой из этих проектов в той или иной степени будет стимулировать развитие IT-отрасли.

Среди национальных проектов следует выделить проект под названием «Цифровая экономика 2024»[3], концепция которого направлена на развитие цифровой отрасли[4]. В частности, одной из целей проекта заявлено: «использование преимущественно отечественного программного обеспечения государственными органами, органами местного самоуправления и организациями», что не возможно без достаточного количества специалистов в области разработки программного обеспечения. Конечно, проблема нехватки специалистов в области информационных технологий осознана уже давно. Меняются подходы в преподавании Информатики в школе, каждый год увеличиваются квоты бюджетных мест по соответствующим специальностям в высших и средних специальных учебных заведениях, появляются курсы программирования для школьников[5], в создании которых непосредственно участвуют крупные корпорации. На наш взгляд это правильные меры, но их недостаточно. Действительно, ведь приход, скажем, большого количества абитуриентов на нужные специальности, это только часть проблемы. Если абитуриенты не готовы к тому, чтобы стать профессионалами именно в этой области, то вряд ли университет сможет сделать из всех нужных специалистов. А наличие диплома у слабо-подготовленного специалиста – это отрицательный фактор и для отрасли, и для самого выпускника, да и для всей социально-экономической обстановки в стране.

Значит назревает вопрос перестройки самого преподавания Информатики в средней общеобразовательной школе. Этот вопрос касается многих аспектов, в частности подготовки учителей информатике, при довольно слабых конкурсах на эту специальность в педвузах. Важно отметить, что должно изменится само содержание данной дисциплины в школе.

---

[1] Эксперты Microsoft предупредили о грядущей нехватке программистов в России и мире – https://infostart.ru/journal/news/news/eksperty-microsoft-predupredili-o-gryadushchey-nekhvatke-programmistov-v-rossii-i-mire_1089181/.

[2] IT-индустрию США ждёт острый дефицит программистов? – https://dev.by/news/it-industriyu-ssha-zhdyot-ostryy-defitsit-programmistov.

[3] Цифровая экономика 2024 – https://digital.ac.gov.ru/.

[4] Национальный проект Цифровая экономика полностью вписывается в концепцию так называемой 4-й промышленной революции.

[5] Одним из таких проектов является в частности Яндекс.Лицей https://yandexlyceum.ru/.

Необходимо повысить роль программирования, изменить содержание, структуру и объем преподавания.

Мы не ставили своей задачей рассмотреть все аспекты проблемы изменения подходов к преподаванию информатики. Мы остановимся лишь на некоторых важных вопросах: место программирования, объем изучения программирования, возраст школьников, когда можно начинать учить программированию, содержание курса программирования.

Общие замечания

Один из пионеров отечественной школы программирования академик А.П. Ершов в одном[6] из своих докладов назвал программирование второй грамотностью, проводя параллели между обычными текстами на естественном языке и программами для машинной обработки. Метафора, использованная автором, не так уж далека от истины, особенно сейчас, когда решения любых вопросов в области экономики, образования, медицины, производства и др. областей не возможно без использования программирования в том или ином виде. В этой связи напрашиваются некоторые соображения, касающиеся преподавания программирования в школьном образовании.

Первый важнейший вопрос, на который следует ответить, это вопрос о содержании школьного курса информатики. В 80-е годы прошлого века в значительной степени под влиянием советской школы программирования [1–3] предмет информатика сводился в значительной степени к обучению программированию. Наверное, не стоит быть в настоящее время столь категоричным, но есть смысл пересмотреть роль программирования в школьном курсе, в том числе и в сторону увеличения количества часов на данную область знаний. Вполне возможно, оставить не большой курс Информатики, как общую дисциплину об информационных процессах, и ввести дополнительный курс, содержание которого будет посвящено различным вопросам программирования, а главное, практике написания программ.

Важный вопрос школьного обучения информатики: с какого года начинать обучение программированию. Конечно, это требует определенной проработки и исследования, но ведь такие предметы, как математика изучаются в школе с первых классов, а не так давно в школах факультативно появились шахматы. Параллель с шахматами проводится нами не случайно, так как программирование как шахматы, как и математические дисциплины также требует большого умственного напряжения, но необходимы ребенку, в частности, и для его интеллектуального развития. Но если шахматы интеллектуальная, но все же игра, то программирование это одна из самых востребованных в настоящее время в мире профессий. И если школьник еще за школьной партой поймет, что из себя представляет этот вид деятельности, то это будет важнейшей профориентационной работой, дающей в конечном итоге значительный вклад в экономику. Опыт, в том числе и авторов данной статьи, показывает, что школьники шестого–седьмого классов могут успешно усваивать азы программирования и решать относительно сложные задачи.

Любые изменение в той или иной сфере ставят вопрос о том, кто будет осуществлять нововведения. Это важный момент, поскольку консервативность образования в значительной степени связана с консервативностью людей, которые играют решающую роль в любом процессе обучения. Мы имеем в виду в первую очередь учителей. К сожалению, профессиональные компетенции значительной части современных учителей информатики формировались, когда программирование в школе еще не приобрело своей значимости.

[6] Выступление А.П. Ершова впервые прозвучало на Третьей всемирной конференции ИФИП и ЮНЕСКО по применению ЭВМ в обучении. Текст доклада на русском языке можно найти в электронном архиве автора по адресу http://ershov.iis.nsk.su/ru/second_literacy/article.

Переход к другой парадигме обучения программированию потребует таким образом переподготовки некоторой части учителей и изменению учебных программ для студентов педагогических вузов по специальности учитель информатики.

Перечисленные три важных проблемы, которые необходимо решать в свете назревшего вопроса об изменения роли программирования в школьном курсе информатики, конечно, не исчерпывают всех задач в этой области. Важнейшей частью изменения должно стать содержимое курсов, на чем мы остановимся в следующих разделах статьи.

Инструментарий обучения

При изучении программирования важно выбрать язык, на котором школьник будет учиться писать программы. Чтобы подойти к этому выбору, следует остановиться на том, в чем заключается профессиональная деятельность программиста, т. е. специалиста, который пишет программный код. Было сломано много копий по поводу того, что такое программирование [4; 5]. Правильный ответ, заключающийся в том, что программирование это инженерная дисциплина (наука) не дает ответ на все вопросы. В частности, на вопрос о том, из каких компонентов состоит эта дисциплина. Можно выделить как минимум две таких составных части (важные для нашего анализа): алгоритмизация (как процесс написание алгоритма) и технология. Под технологией написание программы в данном случае будем понимать совокупность методов и инструментов, имеющих отношение к конкретному языку программирования, с помощью которых создается программа, реализующая конкретный алгоритм.

В повседневной практике в процессе написания программы (или ее части) присутствует и алгоритмический и технологический компонент. При чем у профессионального программиста отделить одно от другого достаточно трудно, ведь для реализации того или иного алгоритма он может использовать специфические средства языка или использовать внешние библиотеки[7]. Невозможность отделения друг от друга данных элементов, при работе одного программиста не означает, однако, что нельзя построить процесс разработки так, чтобы написанием алгоритмов и написанием программ, по готовым алгоритмам занимались совершенно разные люди. Во многих крупных фирмах-разработчиках такое разделение существует. При этом алгоритм пишется на, так называемом, псевдокоде – языке для написания алгоритма, который, обычно не специфицируется и рассчитан на интуитивное понимание человеком. Такое разделение предполагает увеличение скорости разработки и уменьшения количества допущенных ошибок.

Какое отношение это имеет к вопросу о том, как преподавать программирование в школе? Самое прямое. Прежде чем говорить чему и на чем будем учить, следует выбрать приоритеты. Мы имеем в виду выбор между алгоритмической и технологической частями. Возражение этому, что может и ни к чему ставить приоритеты, не работает просто потому, что выбор языка и других инструментов все равно расставляет эти приоритеты. Следовательно, разумно было бы определится с этими приоритетами заранее.

В российских школах долгое время использовали алгоритмический язык Паскаль [6–12], реже Бэйскик [9; 13]. В качестве исключений можно назвать языки C, C++ [14; 15], которые обычно изучается факультативно. Все это компилируемые языки[8]. Традиционно в такого типа

[7] Скажем, вместо того, что самому писать процедуру сортировки массива, можно воспользоваться или готовым методом или процедурой из внешней библиотеки, а для реализации в памяти древовидной структуры у таких языков как C и Python есть совершенно непохожие друг на друга средства.

[8] Для языка Бэйсик существуют и интерпретаторы, и компиляторы.

языках алгоритмической составляющей значительно больше, чем в интерпретируемых языках. Скажем решаются задачи на различные действия с массивами: поиск в массиве, выделение сегментов в массиве, слияние массивов, сортировка массивов и т. п. В таких языках как Python [16–18] в качестве массивов используются такой объект как списки[9], для которых существует целый набор методов, позволяющих в одно действие решать то, что в других языках требует написания отдельной процедуры.

Интерпретируемые языки программирования, к которым, в частности, относится Питон, более располагают к занятиям технологической частью разработки программ. Как правило, в них уже встроены различные структуры, позволяющие технологически решить ту или иную задачу, почти не касаясь глубокого понимания алгоритма. К таким структурам относятся в частности: списки, кортежи, строки, множества, словари. Кроме наличия таких структур в арсенале языка, как правило, имеется набор методов для управления этими объектами[10]. В качестве примера рассмотрим достаточно тривиальную задачу, которую, однако можно решать и с упором на алгоритм и чисто технологически. При этом при использовании, скажем, языка Паскаль придется применить алгоритмическое мышление, а с использованием возможностей языка Питон алгоритмическая часть практически отсутствует.

Задача. С клавиатуры последовательно вводятся три натуральных числа не обязательно с соблюдением какого-либо порядка. Нужно вывести те из них, которые делятся на 3 в порядке возрастания.

Решение 1. Программист на языке Паскаль или С, использовал бы условные конструкции. Вариантов решения несколько, при чем часть из них предполагает вложенные условия. Например, так (опуская всю программу на языке Паскаль):

```
readln(a);
readln(b);
readln(c);
if(b>a) and (b>c) then
begin
        t:=b; b:=a; a:=t;
end;
if(c>a) and (c>b) then
begin
        t:=c; c:=a; a:=t;
end;
if(c > b) then
begin
        t:=c; c:=b; b:=t;
end;
if a mod 3 = 0 then writeln(a);
```

[9] Мы не останавливаемся на различии понятий «массив» и «список», который сводится к разным способам устройства памяти для хранения этих структур.

[10] Как правило, все перечисленные структуры являются объектами в смысле объектно-ориентированной парадигмы.

```
if b mod 3 = 0 then writeln(b);
if c mod 3 = 0 then writeln(c);
```

Решение 2. Решение на языке Питон проще произвести присвоив указанные числа элементам списка и использовав метод сортировки. Данный подход воспроизводится почти на автомате.

```
a = list()
for i in range(3):
 a.append(int(input()))
a.sort(reverse=True)
for t in a:
 if t % 3 == 0:
 print(t)
```

Как видим, решение на языке Питон проще, лишено всяческих алгоритмических изысков, но требует некоторых технологических знаний языка. Важно отметить, что использование технологических возможностей языка позволяет значительно сократить время разработки программы. Но это имеет также и методический смысл: обучаемый быстро начинает создавать достаточно объемные программы, ощущает конкретный результат раньше на языке Питон, чем на языке Паскаль. Разумеется, обучаемому на языке Питон все равно придется столкнуться с необходимостью писать более сложные алгоритмы, но уже после того, как он почувствовал вкус к программированию.

Конечно, вопрос о выборе языка для обучения можно рассматривать в еще более конкретизированной форме. Например, если сравнивать язык Паскаль и язык Питон, не в пользу первого говорит то, что Паскаль довольно редко используется в промышленных разработках. Разумеется, учащийся получает дополнительный стимул, когда он учится писать программы на языке, который широко используется для разработки большого числа приложений.

Конечно, язык является основным инструментом в обучении программированию. Но важным элементом является и среда. Для начинающих программировать среда должна быть простой в усвоении. Например, среда Lazarus, на наш взгляд, является слишком сложной для начинающих программировать на Паскале. Более простой для усвоения является такая программа, как Geany, которую, кстати, можно настроить для работы с разными языками программирования. Аналогично для языка Питон наиболее простой и функциональной в использовании является программа Wing, которую можно использовать, по крайней мере, на начальном этапе обучения.

На наш взгляд пришло время рассмотреть и такой вопрос обучения программированию, как проверка заданий. До сих пор традиционно задания проверял учитель. И, на первый взгляд, это правильный подход. Ведь учитель может проанализировать программу ученика, указать на ее сильные или слабые стороны, определить в чем ошибка. Но есть и другая сторона проблемы. Определить, правильно ли работает программа или нет, путем просмотра ее текста иногда довольно трудно. Тем более проверка ее на двух–трех наборах входных данных не может служить гарантией ее правильности. А если учесть еще тот факт, что работать приходится со многими учениками, то задача для учителя усложняется многократно. Мы считаем, что пришло время широко использовать тестирующие системы для проверки заданий по программированию. Такие системы могут быть установлены на локальные сервер в локальной сети школы, но проще воспользоваться подобными ресурсами в сети Интернет. Наиболее

известным ресурсом для проверки заданий по программированию является Яндекс.Контест[11]. Важно, что преподаватель может создавать свои собственные задачи, определять для них набор тестов, определять время их решения, с возможностью для каждого из учеников набирать баллы. Это в значительной мере освобождает время учителя, которое он может потратить для работы с отстающими учениками, на объяснение решений и т. п. Кроме того, использование такого инструмента нисколько не снижает роль учителя, ведь он всегда может дополнительно проверять и разбирать тексты программ своих учеников. Еще одной важной особенностью подобного инструментария является также возможность контроля времени выполнения программы и, следовательно, рассмотрения такого вопроса, на который мало обращают внимание в школьных курсах Информатики, как оптимизация программ.

О содержании курса

Как было сказано ранее, в преподавании программирования есть две составляющих: технологическая и алгоритмическая. Мы считаем, что наиболее правильный подход должен сочетать в себе оба этих элемента. Реализовать такой подход можно следующим образом:

1. Выбрать язык программирования с высокой технологической составляющей, например Питон.
2. При рассмотрении тех или иных технологических возможностей языка, сделать ударение также и на алгоритмические возможности их реализации.

Остановимся подробнее на некоторых частных вопросах преподавания программирования. Одним из интересных разделов, с которыми сталкивается изучающий программирование на таких языках как Питон или C# это списки. До определенной степени список можно рассматривать как обобщение такой классической структуры как массив, хорошо известной программистам на языке Паскаль и С. Суть обобщения заключается в том, что список в таком языке как Питон изначально рассматривается как объект, с динамически изменяемым размером и возможностью присваивать элементам списка другие структуры, в том числе и другие списки. Наличия большого набора инструментов, с помощью которых можно выполнять различные действия над списком избавляет программиста от написания таких алгоритмов как: поиск элементов списка, поиск списка в другом списке, сортировка списка, удаление и вставка сегментов списка и др. Все эти алгоритмы, как правило, изучаются при изучении такой темы, как «Массивы» в языках Паскаль или С. Мы предлагаем расширить содержание темы так, чтобы ученики могли познакомиться и с тем, как на практике можно реализовать возможности объекта список, в отсутствии некоторых его методов. Примерное содержание темы Списки, ориентированное на язык Питон представлено ниже. Вся темы были разбита нами на шесть подтем.

1. *Введение в списки (list). Создание и заполнение списков. Список как объект. Методы списка. Операции над списком (объединение и умножение на число). Сравнение и присвоение списков.*
2. *Алгоритмы реализации объединения, сравнения, присвоения списков. Поиск элемента в списке. Алгоритм реализации поиска элементов списка.*
3. *Вставка и удаление элементов в списке. Алгоритмы вставки и удаления. Алгоритм поиска списка в другом списке.*

[11] Ресурс Яндекс.Контест располагается по адресу https://contest.yandex.ru/ и требует некоторого времени для регистрации. Определенное время понадобится преподавателю, чтобы освоить не простой, но богатый инструментарий сервера. В качестве еще одного ресурса для проверки задач по программированию укажем также сайт http://contest.uni-smr.ac.ru/ru/ с более простым интерфейсом.

4. *Понятие сортировки. Метода сортировки списков. Алгоритм простейшей сортировки (пузырьковая).*
5. *Понятие среза (slice). Операции над срезами. Примеры алгоритмов реализации операций над срезами (выделение, удаление, замена и вставка среза).*
6. *Смешанные списки – списки, элементами которых являются разные типы данных. Вложенные списки. Двумерные списки – матрицы. Алгоритмы в матрицах (поиск, выделение диагоналей, транспозиция). Ввод и вывод двумерных списков.*

Приведем подробную методическую разработку 1-й подтемы, взятую из разрабатываемых одним из авторов статьи, методических рекомендаций по преподаванию программирования в средней школе на базе языка Питон.

Тема Списки изучается уже после того, как пройдены следующие темы: Переменные (числовые и строковые), Клавиатурный ввод и вывод, Условные конструкции, Циклы и вложенные циклы. К теме Списки предлагается подойти последовательно на основе задач, постепенно усложняя условие. Предлагается решить в начале следующую задачу. У ученика в журнале по данному предмету стоит всего одна оценка (на входе программы). Требуется определить среднюю оценку. Учащие знают, как вычисляют среднее значение и прекрасно понимают, что среднее значение просто совпадет со стоящей в журнале оценкой. Важно, однако, подчеркнуть, что для хранения оценки нам понадобилась одна переменная. Затем предлагается усложнить задачу и рассмотреть случай, когда оценок две. Такая задача вполне понятна ученикам, и они легко с ней справляются. Укажем, также, что для хранения оценок нам понадобилось две переменных, поскольку мы заранее знали, что оценок две. Далее переходим к случаю с тремя оценками. Разумеется, задача легко решается учениками. Следует особо сделать ударение на том, что, во-первых, мы заранее знаем количество оценок, что определяет количество требуемых в программе переменных. Во-вторых, каждый раз мы вынуждены менять структуру программы: количество функций ввода (input()) и формулу вычисления средней величины. В этом месте можно поставить проблему: что будем делать, если количество оценок будет расти. Каждый раз менять количество переменных и структуру программы? Напрашивается использование нового инструмента – новой структуры и алгоритмов ее использования. Таким инструментом является список. Возможное решение задачи, с произвольным количеством оценок будет таким

```
n, li = int(input()), []
for in range(n)
 li.append(int(input()))
s = 0
for i in range(n):
 s = s + li[i]
print(s / n)
```

Отметим, что разделение ввода данных и вычисления среднего важно с методической точки зрения. Более компактное решение задачи можно привести позднее. На примере этой задачи и программы можно легко показать, как можно по-другому задать и перебрать элементы списка.

На наш взгляд очень важно уже заранее ввести для обучаемых такое понятие, как объект. Пока это будет только интуитивный подход. Нам важно указать, что объект обладает таким инструментарием, как методы, которые позволяют выполнять с ним разные

действия. Пример такого метода был использован в программе – append. Можно также перечислить и другие методы, не объясняя подробно их возможности. Кроме собственных методов для списков можно использовать и внешние функции языка Питон, например, функции len, max, min. Третьим механизмом, который можно использовать при программировании списков, это так называемая перегрузка операторов. Следует объяснить понятие термина «перегрузка» – использование одного и того же оператора для объектов разного типа. Речь идет об операторах «+» (объединение списков) и «» (повтор списка), «=» (присвоение списка другому списку), «==» (сравнение списков). Для закрепления данного материала необходимо прорешать несколько не сложных задач. Например, можно продолжить тему классного журнала: «Даны оценки по несколько предметам (списки), необходимо получить общую среднюю оценку».*

После того, как понятие списка будет усвоено, когда ученики узнают какие возможности есть при работе со списками, следует перейти, на наш взгляд, к важному вопросу, рассмотрев, несколько задач на реализацию некоторых встроенных механизмов списков. Например, сравнения списков, их объединение, их присвоение. Здесь учащиеся перейдут на совершенно иной уровень владения материалом, увидев, что все эти возможности они могут реализовать сами и одновременно получат бесценный опыт программирования структур данных.

Баланс между алгоритмической и технологической частями курса программирования можно регулировать как раз на основе алгоритмической части курса, используя три уровня изучения:

1. Ознакомительный. Учащихся знакомят с алгоритмами в общих чертах, говорят об общих принципах их построения.
2. Разбор алгоритма с демонстрацией работы. Приводятся примеры алгоритмов на языке, который изучают учащиеся, объясняется представленный код. Показывают примеры их работы.
3. Решение задач на использование алгоритма. После разбора общих принципов построения алгоритмов, учащимся предлагается решить конкретные задачи, с использованием этих алгоритмов. Также возможно сравнение реализованных алгоритмов и встроенных методов, в частности по скорости выполнения.

Выводы

Нами были рассмотрены вопросы, касающиеся преподавания программирования в школе. Подведем некоторые итоги изложенного в статье.

1. Назрел вопрос изменения отношения к преподаванию программирования в школе. Связано это прежде всего с тем, что роль IT-технологий во всех сферах жизни от количественного роста переходит к качественным изменениям. В ближайшее время в Российской экономике будут реализованы прорывные планы в области цифровой экономики и это потребует подготовки большего количества специалистов в области программирования.
2. Одним из возможных решений задачи, которые стоят перед школой в цепочке подготовки будущих IT-специалистов, является выделение обучения программированию в отдельную дисциплину. Есть смысл определится с тем с какого класса целесообразно начать знакомиться с программированием. Важно также обратить внимание на подготовку и переподготовку учительского состава.

3. Необходимо более подробно разработать парадигму обучения программированию в школе. Сформировать требования к содержанию, языкам и инструментам, которые следует использовать при обучении.

4. При разработке программы курса программирования и конкретной ее реализации следует учесть баланс между технологической и алгоритмической составляющими программирования. Этот баланс должен быть представлен в учебных программах и методиках преподавания программирования.

ЛИТЕРАТУРА

1. Ершов, А.П., Шура-Бура, М.Р. Становление программирования в СССР. Издание второе [ТЕКСТ] / А.П. Ершов, М.Р. Шура-Бура – Новосибирск, 2016, 78 с.
2. Ершов, А.П. Информатика и вычислительная техника. Часть первая. [ТЕКСТ] / А.П. Ершов, В.М. Монахов, А.А. Кузнецов, Я.Э. Гольц, М.П. Лапчик, А.С. Лапчик, А.С. Лесневский, Ю.А. Первин, Д.О. Смекалин – М.: Просвящение, 1985, 96 с.
3. Ершов, А.П. Информатика и вычислительная техника. Часть вторая. [ТЕКСТ] / А.П. Ершов, В.М. Монахов, А.А. Кузнецов, Я.Э. Гольц, М.П. Лапчик, А.С. Лапчик, А.С. Лесневский, Ю.А. Первин, Д.О. Смекалин – М.: Просвящение, 1986, 143 с.
4. Абельсон Х., Сассман Дж., Сассман Дж. Структура и Интерпретация Компьютерных Программ [ТЕКСТ] / Абельсон Х., Сассман Дж., Сассман Дж. – М.: КДУ, 2012. – 608 с.
5. Хант Энди, Томас Дейв. Программист-прагматик. Путь от подмастерья к мастеру [ТЕКСТ] / Хант Энди, Томас Дейв – С.-П.: Питер, 2007. – 270 с.
6. Босова, Л.Л., Босова, А.Ю. Информатика. Учебник для 8 класса [ТЕКСТ] / Л.Л. Босова, А.Ю. Босова. – М.: Бином, 2014. – 157 с.
7. Семакин, И.Г., Залогова Л.А., Русаков, С.В., Шестакова Л.В. Информатика. Учебник для 9 класса. [ТЕКСТ] / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестаков. – М.: Бином, 2015. – 200 с.
8. Угринович, Н.Д. Информатика. 11 класс. [ТЕКСТ] / Н.Д. Угринович. – М.: Бином, 2017. – 272 с.
9. Гейн, А.Г., Сенокосов, А.И. Информатика и ИКТ. 11 класс. [ТЕКСТ] / А.Г. Гейн, А.И. Сенокосов. – М.: Просвещение, 2012. – 338 с.
10. Поляков, К.Ю., Еремин, Е.А. Информатика. Углубленный уровень. 11 класс. В 2-х частях [ТЕКСТ] / К.Ю. Поляков К.Ю., Е.А. Еремин. – М.: Бином, 2013. – 240 с.
11. Ушаков, Д.М., Юркова Т.А. Паскаль для школьников [ТЕКСТ] / Д.М. Ушаков, Т.А. Юркова. – С.-П.: Питер, 2013. – 320 с.
12. Алексеев, Е.Р. Free Pascal и Lazarus. Учебник по программированию [ТЕКСТ] / Е.Р. Алексеев. – Донецк.: УНИТЕХ, 2011. – 438 с.
13. МакГрат М. Visual Basic – программирование для начинающих [ТЕКСТ] / МакГрат М. – М.: Эксмо, 2017. – 192 с.
14. Каширин, И.Ю., Новичков, С.В. От Си к Си++. Учебное пособие для вузов [ТЕКСТ] / И.Ю. Каширин, С.В. Новичков. – М.: Горячая Линия – Телеком, 2005. – 336 с.
15. Дейтел Пол Дж., Дейтел Харви. Как программировать на С: четвертое издание [ТЕКСТ] / Дейтел Пол Дж., Дейтел Харви – М.: Бином, 2009. – 912 с.
16. Майкл Доусон. Программируем на Python [ТЕКСТ] / Майкл Доусон. – С.-П.: Питер, 2018. – 416 с.
17. Бэрри Пол. Изучаем программирование на Python [ТЕКСТ] / Бэрри Пол. – М.: Эксмо, 2019. – 624 с.
18. Mark Lutz. Learning Python. [ТЕКСТ] / Mark Luitz. – O'REILY, 2013, 1540 p.

Pirogov Vladislav Jurievich
Shadrinsk pedagogical university, Shadrinsk, Russia
E-mail: Vladislav-133@yandex.ru

Popova Ekaterina Igorevna
Shadrinsk pedagogical university, Shadrinsk, Russia
E-mail: 978073@mail.ru

Some aspects of teaching programming

Abstract. The article deals with the teaching programming at school. The authors believe that the teaching programming should be single out as a separate discipline. Algorithmic and technological aspects of programming and the balance between them in teaching are considered. Such problems of teaching programming as the content of the course, the age at which to start teaching programming, teacher training are discussed. Such problems of teaching programming as the content of the course, the age at which to start teaching programming, teacher training are discussed. For teaching programming, it is proposed to use one of the interpreted algorithmic languages, such as Python. The article points out certain advantages of Python over compiled languages such as Pascal. The need to automate testing to programs with the help of one of the servers for testing such tasks is emphasized. To a large extent, this allows for standardized program checking and frees the teacher for additional classroom work. The content of the programming course is examined from the point of view of the balance between the technological and algorithmic parts. It is proposed to include in the study of the technological part the study of algorithms that allow replacing certain technological capabilities of the language (sorting, search, etc.). The authors consider in detail one of the topics of the programming course – lists. An example of the content of this topic and a fragment of methodological development of the authors, one of the subtopics (introduction to the lists). The presented fragment describes in detail the sequence of introduction of the concept and the relationship between the technological and algorithmic parts of programming.

Keywords: programming; programming teaching; programming languages; Python programming language; Pascal programming language; algorithms; programming technology