УДК 37.016:004.43

О преемственности в учебных языках программирования

Л.В. Батан

Учреждение образования «Витебский государственный университет им. П.М. Машерова»

Знание основ алгоритмизации и программирования является неотъемлемой частью компьютерной образованности в современном обществе. Цель обучения алгоритмизации заключается в овладении учащимися структурной методикой построения алгоритмов. В рамках исследуемой нами проблемы мы рассматриваем преемственность как в развитии учебных языков программирования (их эволюцию), так и в аспекте обучения основам алгоритмизации и программирования. Цель данной статьи заключается в теоретическом анализе современного состояния учебных алгоритмических языков.

Общее количество языков программирования в мире постоянно растет. Некоторые языки программирования оказались забытыми, но несли идеи для совершенствования других, более популярных. Зачастую разработчики нового языка программирования включают в него как удачные идеи из «забытых» языков (историчность), так и новые достижения программирования (инверсия). Следовательно, изучение истории языков программирования качественно способствует их развитию. Эволюция учебных языков программирования напоминает дерево эволюции видов животных или растений, известных из курса биологии. В этом состоит суть преемственности в учебных языках программирования.

Ключевые слова: *язык программирования, учебный язык программирования, учебный алгоритмический язык, преемственность, историчность, инверсия.*

On continuity in teaching programming languages

L.V. Batan

Educational establishment «Vitebsk State University named after P.M. Masherov»

Knowledge of the basics of algorithmisation and programming is an integral part of computer education in modern society. The aim of teaching algorithmisation is mastering the structural methodology of building algorithms. Within the frames of the problem, which we are studying, we consider continuity both in the development of teaching languages of programming (their evolution) and in the aspect of teaching the basics of algorithmisation and programming. The aim of the present article is theoretical analysis of the modern state of teaching algorithm languages.

The number of programming languages in the world constantly increases. Some programming languages are forgotten though they brought ideas for the improvement of others, more popular ones. Developers of a new programming language often include into it both ideas from «forgotten» languages (historicity) and new advances in programming (inversion). Consequently, studying the history of programming languages qualitatively facilitates their development. The evolution of teaching programming languages resembles the evolution tree of animal and plant species known from Biology course. This is the idea of continuity in teaching programming languages.

Key words: *programming language, teaching programming language, teaching algorithmic language, continuity, historicity, inversion.*

Первые, достаточно примитивные языки программирования возникли сравнительно недавно. Бурное развитие и совершенствование компьютерной техники и технологий в конце XX – начале XXI века способствовало, в том числе, и развитию языков программирования. Сейчас их насчитывается тысячи.

Знание основ алгоритмизации и программирования является неотъемлемой частью компьютерной образованности в современном обществе. Как показала практика, вытеснение изучения этих вопросов из школьного курса информатики и чрезмерное увлечение подготовкой пользователей привели к тому, что при явном улучшении оснащения школ компьютерной техникой уровень общеобразовательной подготовки школьников заметно снизился.

Цель обучения алгоритмизации заключается в овладении учащимися структурной методикой построения алгоритмов. То есть ученики должны научиться использовать основные алгоритмические структуры: следование, ветвление, цикл; уметь разбивать задачу на подзадачи; применять метод последовательной детализации алгоритма. Для этого существуют и хорошо отработаны такие дидактические средства, как разнообразные учебные исполнители алгоритмов. Изучение алгоритмизации в программистском аспекте связано с введением понятий величины и типа величины, константы и переменной, присваивания значения переменной, действия (операции) над величинами, выражения (арифметические, логические, строковые) [1, с. 309].

В рамках исследуемой нами проблемы мы рассматриваем преемственность как в развитии учебных языков программирования (их эволюцию), так и в аспекте обучения основам алго-

ритмизации и программирования. Цель данной статьи заключается в теоретическом анализе современного состояния учебных алгоритмических языков. На наш взгляд, требуют рассмотрения вопросы исторического развития, инверсии и преемственности в языках программирования.

Материал и методы. Материалом данной работы являются существующие языки программирования, в том числе учебные, которые мы проанализировали с точки зрения преемственности, историчности и инверсии. Преемственность в учебных языках программирования нами рассматривалась как с позиции их исторического появления и развития, так и с точки зрения возможной преемственности при изучении второго языка программирования. При анализе эволюции и преемственности в учебных языках программирования была построена соответствующая схема (как один из вариантов преемственности в УЯП). Рассматривая преемственность в обучении как необходимую связь между новым и старым в процессе развития, мы учли тот факт, что преемственность должна также подразумевать отрицание того, что не укладывается в цели обучения.

Результаты и их обсуждение. Первые языки программирования (ЯП) возникли относительно недавно, различные исследователи указывают 20-е, 30-е и даже 40-е годы XX столетия. Программирование было деятельностью, в которую мало кто был посвящен. Первые языки программирования, как и первые ЭВМ, были довольно примитивны и ориентированы на численные расчеты. Обучение программированию проводилось все больше в частном порядке.

Благодаря разработке в 1958–1960 гг. комитетом по языку высокого уровня IFIP языка программирования Алгол появилось представление о программе как о блочной структуре, состоящей из четко описанных и отделенных друг от друга частей. Были выделены структурные управляющие конструкции: ветвления, циклы, что дало возможность описывать логику программы без использования безусловных переходов. Сейчас подобная структура программы кажется чем-то самоочевидным, но на момент появления Алгола все это было заметным шагом вперед. Алгол стал концептуальным основанием многих языков программирования [2]. С появлением микрокомпьютеров впервые со времени создания ЭВМ к ним получили доступ люди, не являющиеся специалистами в области информационной и вычислительной техники. С тех пор алгоритмизация и программирование – составные элементы обучения информатике. Введем некоторые понятия, связанные с обучением программированию.

Язык программирования – это формальная знаковая система, предназначенная для написания компьютерных программ, которые применяются для передачи компьютеру инструкций по выполнению того или иного вычислительного процесса и организации управления отдельными устройствами. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под ее управлением. Алгоритмический язык – это формальный язык, используемый для записи, реализации и изучения алгоритмов. В отличие от большинства языков программирования, алгоритмический язык не привязан к архитектуре компьютера, не содержит деталей, связанных с устройством машины. Таким образом, всякий язык программирования является алгоритмическим языком, но не всякий алгоритмический язык пригоден для использования в качестве языка программирования. Учебный язык программирования (УЯП) – это язык, предназначенный для обучения программированию. Главные требования, которым должен отвечать такой язык, – это простота и универсальность. Чем проще он будет, тем быстрее его освоит новичок. Универсальность УЯП при необходимости позволит ученику легко освоить другой ЯП. Возможности таких языков могут быть ниже, чем возможности полноценных ЯП, но они и не предназначены для масштабной работы. Однако есть примеры учебных языков программирования, впоследствии превратившихся в полноценные, профессионально используемые языки высокого уровня, – это Паскаль, Бейсик.

Развитие и преемственность учебных языков программирования можно проиллюстрировать схематически (см. схему). Однако следует учесть и тот факт, что разработчики языков программирования не всегда идут по прямому пути, исторически, а стремятся включить в новый язык и новые достижения программирования (инверсия). Исполнитель Черепашка впервые появился в языке Лого для обучения детей программированию – многие современные УЯП используют исполнителей как важное дидактическое средство на начальном этапе изучения основ алгоритмизации и программирования. В языке Модула впервые появилась альтернатива процедурам и функциям – модуль, модульное строение программы присуще языку ИнтАл.

Язык Pascal ABC заимствовал из Си удобный для восприятия способ комментирования программного кода. Можно привести и другие примеры. Поэтому схема является одним из вариантов преемственности в УЯП. Как было указано ранее, на сегодняшний день количество языков программирования исчисляется тысячами. Не имея возможности охватить их все, рассмотрим одну из множества ветвей в развитии УЯП.

Одним из первых и достаточно удачных УЯП является язык Лого, разработанный в конце 1960-х годов. В язык Лого впервые было включено понятие **исполнителя** – Черепашки. «Главное методическое достоинство исполнителя Черепашки – ясность для ученика решаемых задач, наглядность процесса работы в ходе выполнения программы» [3, с. 298]. История создания языка Бейсик также связана с поисками путей решения проблемы обучения доступному пониманию техники программирования. Бейсик подвергся жесткой критике за то, что его простота и бесструктурность поощряют порочные методики и привычки программирования, которые способны привести к краху крупных проектов [4]. Современные диалекты и реализации Бейсика являются более структурированными, но язык стал объемнее и сложнее, его освоение требует больше времени и усилий. Еще одной попыткой сделать Бейсик языком программирования для начинающих стала реализация Small Basic, разработанная компанией Microsoft.

Язык Паскаль изначально был разработан как учебный язык программирования для студентов, позволяющий эффективно решать разнообразные задачи. Паскаль, как преемник Алгола, предусматривает блочную структуру программ. Синтаксис языка требует строгого стиля программирования, который способствует уменьшению количества ошибок. Структура программы на Паскале позволяет даже специалисту, не являющемуся ее автором, обнаруживать и исправлять допущенные ошибки и вносить изменения. Возможно, главной причиной популярности языка был именно тот факт, что он способствовал развитию зарождающегося и набиравшего силу движения за структурное программирование [4–6]. Паскаль стал прародителем нескольких, в том числе учебных, языков программирования. В настоящее время широкое распространение в качестве УЯП получила система Pascal ABC, ориентированная на школьников и студентов младших курсов. В системе реализовано такое дидактически важное средство, как работа с исполнителями Чертежник и Робот, что позволяет обеспечивать наглядность на начальных этапах обучения программированию.

В конце 1980-х годов в СССР в качестве средства перехода от более простых языков (в частности учебного языка Робик) к языкам высокого уровня разработан процедурный язык программирования Рапира. Построенный на основе русской лексики, язык использовался в школах для изучения информатики. Язык Рапира предвосхитил некоторые особенности интерпретируемых языков программирования, созданных на полтора-два десятилетия позже (необъявляемые бестиповые переменные, высокоуровневые составные типы данных: кортежи и множества).

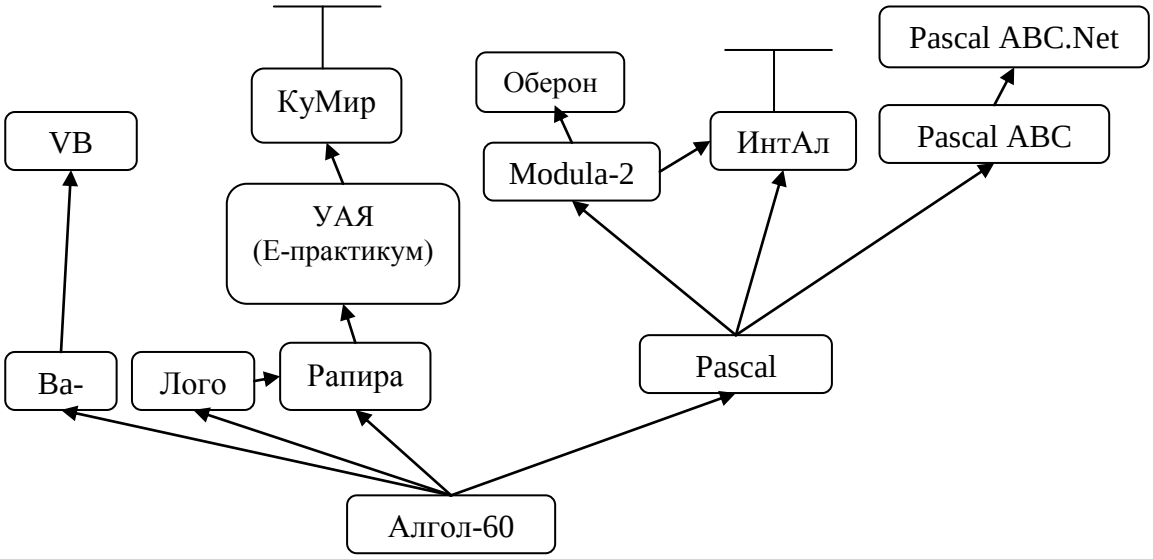

Схема. **Отражение эволюции учебных языков программирования.**

В качестве основы для «безмашинного» курса информатики академиком А.П. Ершовым был введен в употребление алголоподобный школьный алгоритмический язык с русским синтаксисом, впервые опубликованный в учебнике информатики в 1985 г. [7]. Язык также использовался для записи алгоритмов в учебнике «Основы информатики и вычислительной техники» [8]. Для подкрепления теоретического изучения программирования был создан редактор-компилятор Е-практикум, позволяющий создавать программы на алгоритмическом языке. В 1986 г. для Е-практикума выпущен комплект учебных миров (исполнителей) «КуМир»: Робот, Чертежник, Двуног, Вездеход, которые позволили просто вводить понятия алгоритма. Язык этой системы программирования также называется КуМир. Он постоянно дорабатывался и в свое время получил широкое распространение.

В 1996 году в школьной информатике активно использовалась система ИнтАл, поддерживающая работу с несколькими графическими исполнителями и предназначенная для разработки программ на одноименном учебном алгоритмическом языке (УАЯ). Система состоит из основной программы и динамически подключаемых библиотек, каждая из которых содержит полное описание некоторой среды. Благодаря возможности подключения исполнителей среда ИнтАл явилась дидактически важным средством на начальном этапе обучения программированию. Программа на языке ИнтАл состоит из одной или нескольких подпрограмм, что, в частности, указывает на влияние языков Си и Модула-2. Застой в развитии языков КуМир и ИнтАл привел к тому, что в качестве учебного и даже профессионального языка начинает распространяться PascalABC.

Для обучения программированию в ряде случаев используется язык Си. Однако задуманный как профессиональный язык Си непрост для школьников, нуждается в адаптации даже при факультативном изучении в средней школе [9, с. 107].

Общее количество ЯП в мире постоянно растет. На сегодняшний день можно насчитать огромное количество ЯП, разработанных в течение сравнительно небольшого временного промежутка – начиная с 20-х годов прошлого века. Однако не все языки оказались «жизнеспособными». Понимая преемственность как связь между различными этапами или ступенями развития, сущность которой состоит в сохранении тех или иных элементов целого или отдельных сторон его организации при изменении целого как системы [10, с. 380], можно говорить о преемственности в учебных языках программирования. Создание, разработка нового языка программирования или его версии не означают простого забвения старого, иначе было бы невозможно развитие. «Прогресс в развитии обусловливается тем, что сохраняется преемственность между старым, отрицаемым, и новым, утвердившимся. Следовательно, здесь важно уметь найти в старом не только подлежащее отрицанию, но и необходимое для сохранения» [11, с. 9–10]. Как видно из схемы, эволюция УЯП напоминает дерево эволюции видов животных или растений, известных из курса биологии. Менее жизнеспособные языки (ветви) «отмирают», языки, которые более соответствуют реалиям современности, развиваются, в том числе за счет «изюминок», взятых из устаревших языков. В этом состоит суть преемственности в УЯП. Однако преемственность в обучении, как необходимая связь между новым и старым в процессе развития, должна также подразумевать отрицание того, что не укладывается в цели обучения. Примером тому может служить отказ от обязательной нумерации строк в программе на современных версиях языка Basic.

Заключение. История развития языков программирования указывает на примерно 90% отмирания. Некоторые ЯП оказались забытыми, но несли идеи для совершенствования других, более популярных. Налицо преемственность: каждый современный язык программирования примерно на четверть состоит из идей своего предшественника, наполовину соответствует реалиям сегодняшнего дня (служит для решения современных задач), примерно четверть заложенных в нем идей – это идеи, которые будут развиваться в будущем.

Разработчики конкретной версии языка программирования не всегда идут по прямому пути, исторически, а стремятся включить в новый язык как «забытые» удачные идеи, так и новейшие достижения современных языков. Это значит, что в данном процессе присутствует и инверсия и историчность. Следовательно, несмотря на то, что чисто исторически ориентированный порядок изучения ЯП требует больших затрат, изучение истории развития языков программирования необходимо. Об этом говорит и тот факт, что Алгол-60 некоторое время не использовался, был забыт, однако Н. Вирт, «вспомнив» Алгол, «придумал» Pascal.

Историчность и преемственность в УЯП прослеживается также в работе с исполнителя-

ми алгоритмов. Понятие «исполнитель», которое возникло спонтанно [9, с. 55], повлекло «изобретение» целого ряда графических компьютерных исполнителей, являющихся мощным дидактическим средством на начальном этапе обучения программированию. Позже была осознана фундаментальность понятия «исполнитель» в информатике.

ЛИТЕРАТУРА

1. Норенков, И.П. Информационные технологии в образовании / И.П. Норенков, А.М. Зимин. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 352 с.
2. Себеста, Р.В. Первый шаг к совершенствованию: язык ALGOL 60 / Р.В. Себеста // Основные концепции языков программирования. – 5-е изд. – М.: «Вильямс», 2001. – С. 672.
3. Лапчик, М.П. Методика преподавания информатики: учеб. пособие для студ. пед. вузов / М.П. Лапчик, И.Г. Семакин, Е.К. Хеннер; под общ. ред. М.П. Лапчика. – 3-е изд. – М.: Издательский центр «Академия», 2006. – 624 с.
4. Dijkstra, E.W. Go To Statement Considered Harmfu / E.W. Dijkstra // Comm. of the ACM. – March 1968. – Vol. 11, no. 3. – P. 147–148.
5. Wirth, N. A contribution to the development of ALGOL / N. Wirth, C. Hoare // Comm. of the ACM. – June 1966. – Vol. 9. – P. 413–432.
6. Böhm, C. Flow diagrams, Turing machines and languages with only two formation rules / C. Böhm, G. Jacopini // Comm. of the ACM. – May 1966. – Vol. 9. – P. 366–371.
7. Основы информатики и вычислительной техники: в 2 ч. / А.П. Ершов [и др.]; под ред. А.П. Ершова, В.М. Монахова. – М.: Просвещение, 1985.
8. Кушниренко, А.Г. Основы информатики и вычислительной техники / А.Г. Кушниренко, Г.В. Лебедев, Р.А. Сворень. – М.: Просвещение, 1993.
9. Бочкин, А.И. Методика преподавания информатики: учеб. пособие / А.И. Бочкин. – Минск: Выш. шк., 1998. – 431 с.: ил.
10. Философский словарь / под ред. И.Т. Фролова. – 5-е изд. – М.: Политиздат, 1987. – 590 с.
11. Батаршев, А.В. Преемственность обучения в общеобразовательной и профессиональной школе (теоретико-методологический аспект) / А.В. Батаршев; под ред. А.П. Беляевой. – СПб.: Ин-т профтехобразования РАО, 1996. – 80 с.

Поступила в редакцию 25.05.2011. Принята в печать 30.06.2011

Адрес для корреспонденции: 210038, г. Витебск, Московский пр-т, д. 33, УО «ВГУ им. П.М. Машерова», кафедра информатики и ИТ**,** e-mail: larisa@binro.net **–** Батан Л.В.