ЭВОЛЮЦИЯ ПОНЯТИЙ «ПОЛЬЗОВАТЕЛЬ» И «ПРОГРАММИСТ» В СВЯЗИ С РАЗВИТИЕМ И ИЗУЧЕНИЕМ ИНФОРМАТИКИ

БОЧКИНА. И., БАТАН Л.В.

УО «ВГУ им. П.М. Машерова», УО «МГУ им. А.А. Кулешова»
г. Витебск, г. Могилев, Республика Беларусь
E-mail: bochkin.bai@mail.ru, lluckina@mail.ru

В учебной дисциплине «информатика» наблюдаются два взаимодополняющих направления в обучении: подготовка пользователей готовых программ и подготовка программистов. Понятия «пользователь» и «программист» являются важнейшими для принятия верных, обоснованных решений, как при разработке и применении программных средств, так и при определении целей изучения информатики в школе и вузе.

Ключевые слова: программист, пользователь.

Информатика — молодая, но динамично развивающаяся отрасль и учебная дисциплина. С течением времени появилось большое количество прикладных программ, стиль управления и решения задач за компьютером становится все понятнее «неискушенному» пользователю. Поэтому все больше разделов школьной информатики имеют прикладное значение. Наряду с этим можно наблюдать тенденцию снижения возраста обучаемых, изучающих информатику. Тем не менее, с первых лет преподавания информатики в школе в учебную программу по этому предмету входит раздел «Основы алгоритмизации и программирования». Таким образом, мы видим два разных взаимодополняющих направления в обучении школьников информатике. С одной стороны — подготовка пользователей готовых программ, с другой стороны — подготовка программистов.

Понятия «пользователь» и «программист» являются важнейшими для принятия верных, обоснованных решений, как при разработке и применении программных средств, так и при определении целей изучения информатики в школе и вузе. Они прошли (и проходят) через значительную эволюцию, осознание которой должно помочь избежать методических просчетов, доходивших порой до уровня школьных учебников, и для концептуально выверенных решенийпри профессиональном программировании.

Простейшее понимание этой пары понятий сводится к пересказу слов: пользователь — пользуется программами, программист их создает. Но и оно не является исторически первичным.

При работе на ЭВМ первого поколения программист на ассемблере или прямо в машинных кодах решал свои задачи. О пользователе речи не было вообще,программист самиспользовал свои продукты. Ввиду отсутствия текстовой информации не было и комментариев и тем более инструкции по применению

программы. В случае отсутствия или увольненияпрограммистаего программа становилась неприменимой и бесполезной. Малое число ЭВМ и сложность программирования на них делали практически невозможным обучение рядовых студентов и школьников.

СЭВМ второго поколения (Минск, БЭСМ-6, Наири) ситуациянесколько изменилась. Появились языки программирования:Бейсик, Фортран, Кобол, Алгол. Текстыпрограммпрактически не сопровождались комментариями — никому не приходило в голову перфорироватькомментарии. Но уже здесь изза различия между управлением задачей на трансляцию, самой трансляцией и запуском программытермин «программист» раздвоился на прикладное и системное программирование. Прикладной программист, и это важно, стал пользователем по отношению к услугам системного программиста и транслятору применяемогоим языка.Появились фонды программных средств, куда программисты могли передать свою продукцию в виде, например, процедур на языке высокого уровня. Документация на бумаге с описанием параметров этих процедур позволяли использоватьпроцедуры другим программистам, но хорошо владеющим языком разработки. Фактически эти процедуры являлись зачатком языка более высокого уровня. Но обработка текстовой информации кроме как трансляторамипрограмм оставалась экзотикой.

Ситуация коренным образом изменилась с появлениемЭВМ третьего поколения. Здесь было развито (увы, до непонятности) системное программирование. Позиция «системный программист» была четко определена. Была поставлена (и достигнута)ложная цель — максимально загрузить процессор задачами разной сложности. О том, что прикладной программист — тоже пользователь, было забыто, и прикладные программисты погрузились в язык низкого уровня —язык управления заданиями.

В связи с появлением дисплеев настал, наконец,черед текстовой информации икомментариев к тексту программ разработчика. Следующий важнейший шаг — меню на экране, позволил, наконец, скрыть язык реализациипрограммы и термин «пользователь» приобрел общепринятый тогда смысл. Отметим, что меню — это для пользователя фактически язык более высокого уровня, чем язык разработчика, и при этом он может быть более простым и понятным. Последующие нововведения (радиокнопки, наборные меню) просто развили эту базовую возможность меню. Из менюпользователь просто что-то готовое выбирает,не вспоминая ни слова из языка реализации.

Персональные ЭВМ, если не учитывать этап провального отказа и отката на них от программных средств ЭВМ третьего поколения, сделали актуальной школьную информатику.Но первые школьные учебники из-за бедности программного обеспеченияэтих персоналок фактически строились как учебники по алгоритмизации и программированию. А.П.Ершовым был предложен полемичный лозунг «Программирование — вторая грамотность». Про пользователей на время опять забыли.

При переходе на более мощные ППЭВМ и накопление разнообразного программного обеспечения к ним и, что важно, развитие интерфейса, стала очевидной необходимость учета этого и в школьной информатике. Решение свелось, увы, снова к очередному полемичному лозунгу: долой программирование, надо готовить пользователей.При этом была совершена концептуальная ошибка в связи с недопониманием эволюции понятий «программист» и «пользователь» с ростом возможностей программного обеспечения.

Прежнее понимание этих категорий основывалось на фактически двухуровневом отношении человека и компьютера. Но еще на ЭВМ третьего поколения уровни программистарасщепились на «системный иприкладной». На ППЭВМ рост разнообразия уровней и мощности программного обеспечения привел к появлению важнейшего для успешности применения ЭВМуровня: «парапрограммист» или «кентавр»— программист, понимающий по сути проблему пользователя и решающий ее на языке сверхвысокого уровня: электронные таблицы, СУБД, пакеты типа MAPLE и так далее.

Под давлением появившихся возможностей программного обеспечения разных уровней расщепились и пользователи. Кроме пассивного, неинтересного для нас пользователя, не работающего за ПЭВМ(клиент на вокзале или в банке), появился параметрический, образно говоря, чистый пользователь. В силу высокойпонятности нового программного обеспечения появился и пользователь, не избегающий немного попрограммировать, чтобы улучшить «среду своего обитания»,хотя бы на уровне своего меню (записи макрокоманд), не обращаясь к программистам. Заметим,что и системный администратор в ВУЗе ведет себя просто как параметрический пользователь, задавая пароли и права доступа для студентов.

Это размножение уровней подчеркнуло,что сегодня программист и пользователь — уже не два уровня владения компьютером,а два стиля,две роли человека за компьютером. И тот, кто считает себя программистом, является одновременно и пользователем по отношению к своей системе программирования, например на Паскале или Си.И пользователь может сегодня записать макрос (хотя бы линейный алгоритм) из команд своего языка высокого уровня (созданного программистом) и стать волей-неволей немного программистом.

Чтобы все же различить стиль пользователя и программиста,заметим, что алгоритм, созданный программистом, исполняет компьютер, а алгоритм пользователя исполняет он сам. Но при этом и программист и «слегка программирующий» пользователь проходят через этапы: алгоритм в уме — для себя (внутренняя речь), алгоритм «для другого» (человека, внешняя речь)и алгоритм «для иного» (компьютера, текст на языке реализации).

Осталось определить, чем являются комментарии к тексту программы. Их очень не любят писать ни пользователи, ни программисты любыхуровней.Это можно понять, если заметить, что комментарии — это алгоритмна языке сверхвысокого уровня — уровня постановки задачи, подмножестве живого языка.

Нельзя, например,считать комментарием к команде «S=0‘Положить S равным нулю». Это беспомощная тавтология. Комментарием же, например, является «S=0 ‘Подготовить накопитель для суммирования ряда’»

Парадоксальный признак компетентности программиста — избегать программирования, особенно в эпоху Интернет, использовать готовые программы, не бояться быть пользователем, повышать эрудицию, компьютерную образованность.Парадоксальный признак компетентности пользователя — не бояться немного попрограммировать на своем языке сверхвысокого уровня, хотя бы создавать линейные алгоритмы там, где это позволит сделать более эффективным решенее поставленной задачи. Фактически он уже сам является исполнителем своих линейных алгоритмов, переходя по вложенным меню, и линейный алгоритм для него вполне понятная и знакомая вещь. Осталосьотделить алгоритм от себя, создав и запустив макрос.

Вопросы, связанные с преподаванием информатики в школе впервые были сформулированы во второй половине 20 века. Однако за сравнительно небольшой промежуток времени было разработано большое количество программ и учебников по информатике. При анализе данной учебно-методической литературы нами было замечено различие в подходах к изложению материала у авторов учебников.

При упорядочении тем курса информатики в большинстве школьных учебников использован один из двух подходов: исторический или инверсный. Исторический подход во многом повторяет порядок появления соответствующего материала в реальной информатике. Инверсный — противоположен ему: в первую очередь изучается то, что ближе человеку (а оно позже появилось в информатике), например, обработка графической информации; затем письмо, счет, и, возможно, двоичные числа [1].

Инверсный подход состоит в последовательном изложении материала от простого к сложному. Здесь ученик как бы «плывет по течению», выполняет упражнения и решает задачи согласно накануне разобранного шаблона. Что касается задач, связанных с программированием, то ученик (а часто и учитель) не особо задумывается (часто на рефлексию попросту нет времени) о том, почему появились те или иные методы решения, для какого класса задач они применяются. Данный подход позволяет «натаскать»на решение стандартных задач практически любого ученика, не зависимо от наличия у него математических способностей—это лишь вопрос времени. Современный «средний» ученик ожидает получить знания в готовом виде. И его ожидания отчасти оправданы. Многообразие учебных предметов (дисциплин), насыщенность учебных программ, большой объем материала — все это требует значительных усилий и затрат времени на учебу. Однако, как известно из психологии [2,с.72], знания, полученные в готовом виде, без труда, без умственных усилий, не являются «прочными». Нам видится целесообразным применение инверсного подхода в преподавании тех разделов информатики, которые связаны с освоением инфор-

мационно-коммуникационных технологий и ориентированы на подготовку учеников как будущих пользователей. Это связано с тем, что для освоения большинства приемов и технологических операций глубоких познаний (особенно из истории информатики) не требуется.

Освоение же алгоритмизации, программирования и основ управления в большей степени предполагает системно-комбинаторный подход, умение предвидеть последствия своих действий и команд, выбирать оптимальные решения задач. При изучении данного раздела высоких результатов можно ожидать при использовании подхода противоположного инверсному — исторического. Исторический подход можно охарактеризовать как проблемный метод изложения материала. Данный метод позволяет активизировать мыслительную деятельность учеников, что способствует глубокому усвоению материала. Однако зачастую он требует больше времени, высокого уровня мотивации учащихся и особого математического склада ума.

Таким образом, понимание эволюции понятий «пользователь» и «программист», сознательный синтез деятельности человека в этих крайних, взаимодополняющих ролях должен помочь избежать очередных дидактических ошибок, более взвешенно строить и профессиональную и учебную деятельность с применением ЭВМ.

ЛИТЕРАТУРА

1. Бочкин, А.И. О месте оператора GOTO при изучении программирования / Бочкин А.И., Кузьмичев Д.Р.//Информатика и Образование М.: — 2008, — №4
2. Дьюи, Д. Психология и педагогика мышления. / Д. Дьюи // Пер. с англ. Н. М. Никольской. – Москва : Совершенство, 1997. – 208 с.