**МОУ Уршельская средняя общеобразовательная школа**

**ПРОБЛЕМЫ И ВОПРОСЫ ПРЕПОДАВАНИЯ ПРОГРАММИРОВАНИЯ В ШКОЛЕ**

**Подготовила: Круглова О.Н.**

**учитель информатики и**

**математики Уршельской сош**

**2010г.**

*Имеет значение не багаж (образование),   
 а орудие, которым его схватываешь...   
 А. Экзюпери*

При преподавании курса программирования в школе существует множество проблем, вопросов. Прежде всего, рассмотрим объективные условия преподавания информатики в массовой школе, цели и задачи преподавания, доступные средства и другие вопросы. Без этого нельзя сформировать целостный и систематический курс программирования, являющийся неотъемлемым компонентом школьной информатики и отвечающий потребностям среднего образования и общества в целом.

**Объём учебной нагрузки**

Официальная точка зрения такова: информатика должна преподаваться по одному часу в неделю. Большинство руководителей от образования не вникают в последующие фразы инструктивно-методического письма, о том, что: "при наличии условий объём учебной нагрузки может быть увеличен:" Это положение существенно ограничивает возможности школьных преподавателей. По часу в неделю обычно преподаются дисциплины необязательные, второстепенные, служащие для "отдыха" школьников от "тяжёлых" предметов и для разбавления аттестата хорошими отметками.

Положение не спасает и удвоение учебной нагрузки — её всё равно не будет хватать, уж слишком сложен и объёмен учебный материал даже при изучении одного языка программирования. А ведь школьный курс информатики - это не только программирование!

**Цели и задачи курса программирования**

Многие проблемы в преподавании информатики связаны с нечётким целеполаганием. Основной целью изучения образовательной области "Информатика" считается подготовка школьников к практической деятельности, а развитие мышления и формирование основ научного мировоззрения как бы отступают на второй план.

Изучение программирования позволяет внести свой вклад в достижение этих целей. В первую очередь следует рассматривать программирование как средство развития мышления школьника. Профессиональной подготовкой занимаются специализированные учреждения. Поэтому лучше сразу отказаться от невыполнимой задачи: изучить язык программирования, каким бы простым он не был. Знание языка как и владение любым другим инструментом само по себе ничего не даёт, куда важнее умение им пользоваться, а на это, как правило, времени и не остаётся (причём, это общая проблема школьного образования). Только при самостоятельном решении задач можно говорить о развитии у школьников способности принятия решения и ответственности за его последствия.

На этапе становления информатики казалась вполне логичной мысль о том, что, выучив язык, учащиеся начнут применять его в своей повседневной учебной и исследовательской деятельности. Сейчас эта ниша занята программными пакетами, относящимися к категории новых информационных технологий. Таким образом, идея практического применения полученных знаний по отношению к программированию стала неактуальной. Это означает лишь одно: программирование напрямую не связано с дальнейшей практической деятельностью будущего гражданина и в рамках этого курса не имеет смысла говорить о третьей цели, провозглашённой проектом стандарта — подготовке школьников к практической деятельности. Здесь она может упоминаться разве что в контексте формирования элементарных навыков работы на ПК.

**Преемственность школьного и вузовского курсов информатики**

Стремление соответствовать требованиям времени приводит к мощному прессингу со стороны родителей и широкой общественности в целом. И это понятно, ведь, родители озабочены адаптацией вчерашних школьников к условиям "взрослой жизни". Это, в свою очередь, инициирует стремление школьных коллективов к профессиональной подготовке выпускников — задаче, школе не свойственной. Но с упорством учителя "готовят к поступлению в ВУЗы" старшеклассников, пытаясь поместить в ограниченный учебный план школы институтские курсы программирования. Повсеместная распространённость этого явления привела к тому, что ситуация уже не считается ненормальной. К примеру, значительная часть выпускников не понимают смысла операции присвоения, хотя с ними "проходили" три способа сортировки массивов.

Вузовские преподаватели, в большинстве, негативно отзываются об этих попытках, так как вчерашние выпускники, поверхностно ознакомившись с основами языка в школе, без должного старания относятся к его изучению в институте, что приводит в итоге к академической неуспеваемости. Поэтому среди преподавателей ВУЗов бытует устойчивое мнение: "В школе учить программированию не нужно!"

Отказ от дублирования содержания вузовского обучения, перенос акцентов на выработку понимания основ программирования позволит добиться преемственности среднего и высшего образования. Разделить между ними этапы формирования понятий и закономерностей труднее, чем поделить на "школьный" и "институтский" списки тем и разделов, но это единственно возможный путь для полноценного образования личности.

**Выбор языка программирования.**

В государственном стандарте по информатике отмечается, что в результате изучения информатики и ИКТ на базовом уровне ученик в области программирования должен:

1. знать основные свойства алгоритмов, типы алгоритмических конструкций: следование, ветвление, цикл, понятие вспомогательного алгоритма;
2. уметь использовать алгоритмические конструкции, выполнять и строить простые алгоритмы, выполнять базовые операции над объектами: цепочками символов, числами, списками, деревьями;
3. использовать приобретенные знания и умения в практической деятельности и повседневной жизни при выполнении индивидуальных и коллективных проектов, в учебной деятельности, в дальнейшем освоении профессий.

Данные знания, умения и навыки формируются при изучении темы «Алгоритмизация и программирование».

При построении обучения учащихся теме «Алгоритмизация и программирование» каждый учитель информатики сталкивается с огромным количеством вопросов: как построить изложение материала, какие использовать методические разработки, в какой форме проводить занятия, какие составить практические задания, какой материал использовать учащимся при изучении и другие. Все эти вопросы возникают из-за отсутствия четко и в полном объеме изложенных учебно-методических материалов для изучения данной темы.

Перед началом обучения учителю необходимо выбрать язык программирования с учетом интересов учащихся, их направленности и структуры образовательного процесса в школе. Безусловно, в начале обучения необходимо изучать алгоритмический язык, что является основой для формирования алгоритмического мышления, для понимания и правильного построения алгоритмических конструкций. Но в последнее время уже наблюдается тенденция перехода обучения от алгоритмических языков к объектно-ориентированным языкам программирования, что показывает об изменении общего подхода к преподаванию программирования в школе. Так в учебниках Н. Д. Угриновича «Информатика и информационные технологии. Учебник для 10-11 классов» и «Информатика и ИКТ. Базовый курс: Учебник для 9 класса» изучается тема «Алгоритмизация и программирование» на основе объектно-ориентированного языка программирования Visual Basic.

Каждый школьный учебник по информатике включает в себя различные разделы, связанные с изучением информационно-коммуникационных технологий и основ информатики. В различных учебниках тема «Алгоритмизация и программирование» не рассмотрена достаточно полно, глубоко и доступно, и предполагается разное количество часов на изучение данной темы, а изложенный материал в научных пособиях часто не поддается изучению учащимся. Поэтому задача учителя в школе состоит в разработке такой методики, которая максимально упростит развитие способности программировать.

Я предлагаю построить обучение теме «Алгоритмизация и программирование» последовательно таким образом, чтобы учащиеся на начальной стадии обучения ознакомились с различными языками программирования, смогли понять необходимость изучения алгоритмического языка программирования. В начале изучения учащиеся должны ознакомиться с основными алгоритмическими конструкциями, командами языка, правилами описания объектов языка программирования, структурой программы и правилами написания. Учащимся необходимо сначала сформировать навыки написания простейших программ с использованием алгоритмических конструкций и основных объектов языка программирования, а затем перейти к изучению простых и далее более сложных методов программирования.

**Несоответствие знаний и умений по теме «Алгоритмизация и программирование» , предусмотренных школьной программой требованиям к выпускнику по форме ЕГЭ.**

Ещё одной проблемой сегодня в данной области является несоответствие темы «Алгоритмизация и программирование» в плане экзаменационной работы ЕГЭ по информатике федеральному компоненту государственного стандарта общего образования.

В плане единого государственного экзамена 2010 года включаются следующие проверяемые элементы содержания обучения:

1. умение прочесть фрагмент программы на языке программирования и исправить допущенные ошибки,
2. умения написать короткую (10-15 строк) простую программу обработки массива на языке программирования или записать алгоритм на естественном языке,
3. умение построить дерево игры по заданному алгоритму и обосновать выигрышную стратегию,
4. умения создавать собственные программы (30-50 строк) для решения задач средней сложности.

Все эти элементы включены в блок «C», который считается наиболее сложной частью контрольно-измерительного материала по ЕГЭ. Данные умения требуют от учащихся 11 классов, оканчивающих обучение в школе, глубокое знание темы «Алгоритмизация и программирование», хорошее владение хотя бы одним из языков программирования: Бейсик, Паскаль и т.д.

На учебный предмет «Информатика и ИКТ» в федеральном базисном учебном плане в 8-х и 9-х классах отводится 105 часов (35 учебных часов из расчета 1 учебный час в неделю в 8 классе и 70 учебных часов из расчета 2 учебных часа в неделю в 9 классе). Из этого количества часов отводится 19 часов на изучение темы «Алгоритмы и исполнители», причем подразумевается изучение формальных исполнителей алгоритмов. Среднее (полное) общее образование базового уровня включает в себя 35 часов в 10 классе и 35 часов в 11 классе (из расчета 1 учебный час в неделю). В данное количество часов не входят часы на изучение темы «Алгоритмизация и программирование». Предполагается, что учитель будет использовать язык программирования во время решения задач при изучении других тем.

Таким образом, объём часов на изучение темы «Алгоритмизация и программирование» не дает возможности в полной мере изучить данную тему в школьном курсе. В этом и заключается несоответствие выделяемого количества часов на изучение данной темы с объемом рассматриваемого материала за данное количество часов, и в этом выражается несоответствие к требованиям выпускника по форме единого государственного экзамена.

Некоторые вузы в этом году требуют сдачи ЕГЭ по информатике при поступлении на технические специальности .Для этого учащимся необходимо получить высокие баллы сдачи ЕГЭ, что требует обладания глубокими знаниями и хорошими навыками программирования, чтобы уметь решать задачи блока «С».

**Олимпиады по программированию.**

На сегодняшний день актуально стало олимпиадное движение. Олимпиада по информатике предполагает решение учащимися олимпиадных задач по программированию. Роль олимпиад значительно увеличилась, так как после отмены льгот медалистов при поступлении в вузы, победа в олимпиадах является самым главным показателем качества знаний учащихся, что дает им право на льготное поступление. Поэтому учителям информатики необходимо обучать программированию, чтобы направлять детей к участию в олимпиадах по информатике.

Мне думается что организаторы олимпиад просто немного опаздывают с реакцией на изменения внешних условий. Соревнования по программированию неизбежно займут место среди высокоспециализированных конкурсов типа музыкальных, художественных или спортивных (имею ввиду профессиональные конкурсы).  
Мне кажется что содержание общешкольных олимпиад по ИТ - рано или поздно будет содержать вещи типа Интернет-поиска, обработки и компиляции разноплановой медиа-информации, решение на компьютере очень прикладных задач типа подсчетов сложных процентов по кредиту стандартными вычислительными пакетами - что-то в этом духе.

**Систематическое построение курса**

Несмотря на активную критику и формальное неприятие Бейсика, большинство задач по программированию одинаково решаются — строка в строку — на трёх "официальных" языках: Бейсике, Паскале, школьном алгоритмическом языке. А это свидетельствует о механическом переносе задач, придуманных под Бейсик, в более поздние разработки. Даже в новом обязательном минимуме наряду с упоминанием об объектном программировании присутствует фраза: "Знакомство с одним из языков программирования. Переменные величины: тип, имя, значение. Массивы (таблицы) как способ представления информации".

Интересно, что попытки реализовать курс программирования в одной-единственной, пусть даже самой лучшей программной среде, приводят к значительным методическим затруднениям. Изучение же нескольких учебных сред — роскошь, которую могут позволить себе далеко не все преподаватели.

Одним из путей решения проблемы может стать структурно-модульное разделение курса программирования на три уровня: пооперационный, процедурный, объектный. Каждый уровень может быть представлен собственным программно-методическим комплексом, изучение его может быть формально независимым от изучения других модулей. В зависимости от направленности всего курса информатики (уровень А и уровень Б в терминологии Министерства образования) отдельные модули могут быть расширены или, напротив, вообще исключены из курса.

**Содержание курса программирования**

Разделение всего курса на уровни, с одной стороны, позволит использовать наиболее удобные средства и приёмы для изучения понятий, характерных именно для этого уровня, с другой же стороны, потребует дополнительных усилий для связывания в единый комплекс отдельных модулей, реализующих каждый уровень.

Модульное построение позволит изучать программирование в любом порядке следования модулей и даже отказаться от изучения одного или двух из них в зависимости от конкретных условий преподавания. Построение школьного предмета не обязано повторять исторический путь изучаемой дисциплины, поэтому объектный или процедурный уровень может быть освоен раньше пооперационного. В любом случае, систематизация курса необходима.

*Пооперационный уровень*

Исторически первый уровень. В основном, школьные курсы реализуют именно его. Здесь от учащихся требуется запоминание элементарных команд, которые изменить невозможно. Важнейшие понятия — линейное следование и операция присвоения. Здесь же изучаются алгоритмические структуры, точнее, их внутреннее строение. При этом выясняется, что все структуры можно свести к одной — условному переходу. На данном уровне алгоритм может быть представлен в виде блок-схемы. Этот этап обязателен, а в случае нехватки учебного времени, когда, например, информатика преподаётся в старших классах по часу в неделю, именно им и следует ограничиться.

Основной способ проверки усвоения — трассировка алгоритмов. Во многих курсах предлагается другой способ. Учащимся предлагается решить одну из типовых задач, то есть воспроизвести по памяти текст программы. На первый взгляд, трассировка алгоритма кажется слишком простой задачей, другое дело — составление программы. Но возможно ли с уверенностью утверждать, что учащийся понимает смысл основных понятий и поэтому составил верно исполняющуюся программу? Ведь он вполне мог заучить её текст. При трассировке алгоритма ситуация совершенно другая. Задачи столь похожи друг на друга, а ответ так сильно зависит от исходных данных и порядка операций, что проще решить задачу, чем заучить правильный ответ. В любом случае, прямое измерение по системе "зачёт-незачёт" предпочтительнее любых косвенных оценок.

С изучением пооперационного уровня программирования связана и ещё одна застарелая проблема, а именно "наглядное" представление алгоритмов в виде блок-схем. Несмотря на справедливую критику (см. например Симонович С. и др. "Практическая информатика"), блок-схемы кочуют из одного обязательного минимума в другой и активно используются в обучении программированию, хотя практические программисты их давно не применяют. Похоже, что такой способ повышения наглядности применим только к очень простым алгоритмам, реализуемым на пооперационном уровне. Главное, что в блок-схемах цикл или ветвление представляются в виде отдельных команд, а не единых, логически целостных блоков. Далеко не каждый преподаватель обводит на блок-схеме цикл или ветвление прямоугольной рамкой, подчёркивая такую целостность. Поэтому лучшим выходом представляется выделение блок-схемам своего рода "экологической ниши", они годятся для представления внутреннего устройства стандартных алгоритмических конструкций, но для описания сколько-нибудь завершённой задачи их применять не следует.

Выбор конкретной программной среды для изучения этого этапа — дело учителя, однако количество терминов и понятий, которые следует выучить, прежде чем начать собственно обучение, в среде QBASIC минимально. Конечно, TurboPascal — среда более престижная, но на этом этапе каких-либо преимуществ не даёт. Поэтому, если не предполагается использование этой среды для изучения следующего (процедурного) этапа — лучше остановить выбор на более простом QBASIC.

*Процедурный уровень*

Процедурный уровень интересен тем, что направлен на получение конечного результата, тогда как на пооперационном уровне, в основном, создаются фрагменты программ. Именно с этого уровня начинается структурное программирование. Здесь по-прежнему важно линейное следование, но отдельные команды практически равноправны, а цикл или ветвление представляется как одна операция. Здесь изучается вложенность процедур, в том числе самовложенность (рекурсия). Здесь же вводится понятие параметра как переменной части процедуры, определяющей результат. Текст программы достаточно нагляден даже без применения дополнительных средств.

В ВУЗах традиционно структурное программирование изучается в среде TurboPascal. Использование этой среды приводит к объединению пооперационного и процедурного уровня в одном курсе. При этом знакомство с процедурами происходит в середине периода обучения. В качестве основы для программирования используются, по преимуществу, математические задачи. Например, при изучении рекурсии любимой задачей почти всех авторов, становится вычисление факториала числа. Обычно авторы оправдывают использование не изучаемого в школьной математике понятия простотой его определения. Есть и другие примеры включения в подобные курсы задач, которые лучше иллюстрируют конструкции и возможности языка, а не способствуют лучшему усвоению базовых понятий. Совершенно очевидно, что модуль, реализующий процедурное программирование и рассчитанный на изучение в 7-8 классе, не может опираться на математическое содержание, особенно когда предполагается самостоятельная творческая деятельность учащихся.

Бесспорным лидером в этом модуле являются курсы, основанные на Лого-технологиях. Метод последовательной детализации алгоритма используется как рабочий инструмент, а не только как объект изучения. К тому же изучать вложенность процедур можно практически с самого начала курса, сразу сделав её основным приёмом работы, приобретая заодно привычку структурировать решение любых задач.

В Федеральном перечне имеется только один учебник, посвящённый Лого-технологии, это "Практикум по программированию в среде Logo-Writer" А. Юдиной. Зато исполнители представлены шире: набор исполнителей в "Алгоритмике" А. Звонкина, кочующие из учебника в учебник Робот А. Кушниренко и Паркетчик А. Гейна.

*Объектный уровень*

Этот уровень самый молодой и наименее разработан. Объектный подход стал широко применяться в программировании уже после того, как информатика стала школьным предметом. Появляющиеся в периодической печати и всемирной сети Интернет публикации посвящены, в основном, попыткам знакомства школьников с Delphi или VisualBasic. До сих пор этот уровень воспринимается как экзотический и необязательный для массовой школы. Однако только этот уровень направлен на создание полноценных проектов. Именно его изучение позволит ликвидировать в сознании школьника понятийную пропасть между программами, с которыми он постоянно имеет дело и так называемыми программами, которые ему приходится писать на уроках информатики. Если для учительства связь очевидна, то дети зачастую недоумевают: что общего между яркой, динамичной, интерактивной игрушкой и чёрным экраном с белыми цифрами?

Объектный подход интересен отсутствием явной линейной последовательности в тексте программы, характерной для предыдущих уровней. В самом деле, если программа составлена в категориях объектов, их свойств и событий, трудно говорить о каком-то явном алгоритме, ведь неизвестно, какое событие может произойти в следующий момент. Здесь имеется возможность обсудить понятия, которые не рассматривались на прежних уровнях.

**Вывод.**

Проблема преподавания программирования в обычных школах действительно существует. Дело в том, что по программе его вообще нет или чисто символически. С одной стороны обучение навыкам программирования является наиболее важной частью обучения информационным технологиям так как на всех олимпиадах, конференциях это требуется и для победы, а за частую вообще для участия школьник должен обладать довольно высоким уровнем программирования , а с другой стороны школы соревнуются в том, что наиболее важно, но в школах не ведется!

Я думаю, в чистом виде программирование интересует небольшую категорию людей.. Сегодня простому пользователю программировать не нужно.  
Зачем всех подряд учить программированию, если это реально нужно нескольким ученикам собравшимся в технический вуз причем на соответствующие специальности?!  
 В результате данного исследования я пришла к выводу, что в общеобразовательном классе необходимо ознакомительно изучить какой-нибудь один алгоритмический язык (Basic, Pascal ). Изучение данного языка необходимо для формирования знаний, умений и навыков программирования, а также для формирования абстрактного, логического и алгоритмического мышления у учащихся, а в классах профильного уровня учащиеся могут изучать несколько языков программирования, включая объектно-ориентированные языки (Delphi или Visual Basic).

Литература

1. Аркадьев, А.Г., Днепров, Э.Д. Сборник нормативных документов. Информатика и ИКТ / сост. А.Г. Аркадьев, Э.Д. Днепров. – М.: Дрофа, 2007.
2. Лапчик, М.П. Методика преподавания информатики: Учебное пособие для студентов педагогических вузов / М.П. Лапчик, И.Г. Семакин, Е.К. Хеннер – М.:Издательский центр «Академия», 2003.
3. Окулов, С. М. Задачи по программированию / С.М. Окулов, ТВ. Ашихмина, Н.А. Бушмелева и др. – М.:БИНОМ. Лаборатория знаний, 2006.
4. Угринович, Н.Д. Информатика и информационные технологии. Учебник для 10-11 классов / Н.Д. Угринович. – М.: БИНОМ. Лаборатория знаний, 2007.
5. Угринович, Н.Д. Информатика и ИКТ. Базовый курс. Учебник для 9 класса / Н.Д. Угринович. – М.: БИНОМ. Лаборатория знаний, 2007.