Федеральное агентство по образованию

ГОУ ВПО ТГПУ им. Л.Н.Толстого

Кафедра информатики и вычислительной техники

Курсовая работа по теме:

**Методические особенности изучения раздела "Алгоритм и исполнители" в базовом школьном курсе информатики**

Выполнил:

студентка 5 курса группы “Б”

факультета МФиИ

Федосеева Т.А.

Проверил:

Даниленко С.В.

Тула 2008г.

**Оглавление**

Введение

1.Теоретические основы раздела «Алгоритм и исполнители»

* 1. Определение основных понятий
  2. Формы представления алгоритма
  3. Учебный алгоритмический язык
  4. Решение трудностей изучения раздела учащимися

2. Методические особенности изучения раздела «Алгоритм и исполнители»

2.1Содержание раздела в стандарте

2.2 Обзор авторских программ

2.3 Методические особенности изучения раздела

Заключение

Библиографический список

**Введение**

Раздел «Алгоритм и исполнители» в базовом школьном курсе информатики» занимает одно из центральных мест. Однако до сих пор нет достаточно определенного мнения в вопросах, касающихся более частных аспектов преподавания этого раздела.

Изучение алгоритмизации в школьном курсе информатике может иметь два целевых аспекта: первый – развивающий аспект, под которым понимают развитие алгоритмического мышления учащихся; второй – программистский аспект, под которым понимают развитие навыков составление учебных программ. Первый аспект связан с усилением фундаментальной компоненты курса информатики. Ученикам даются представления о том, что такое языки программирования, что представляет собой программа на языках программирования, как создается программа в различных средах. Второй аспект носит профориентационный характер. Профессия программиста в наше время является достаточно распространенной и престижной. Изучение программирования в рамках школьного курса позволяет ученикам испытать свои способности к такого рода деятельности.

Проблемы изучения раздела «алгоритм и исполнители» связаны прежде всего с большим объемом материала и значительными трудностями для школьников, как с определением понятия алгоритма, построения алгоритмических конструкций, так и при работе с формальными языками программирования, их правильном применении. В следствии с этим возникает множество вопросов, связанных непосредственно с преподаванием этого раздела относительно содержания, методов и средств, используемых для этого.

Различные авторы школьных программ по курсу «Информатика и ИКТ» по разному подходят к построению алгоритмической учебной линии. В том числе по иному решаются проблемы содержания теоретической и практической составляющей разработчиками школьных программ, к ним относятся Н.В. Макарова, Н.Д. Угринович, И.Г. Семакин и другие.

В связи со всем вышесказанным в данной курсовой работе была рассмотрена тема «Методические особенности изучения раздела «Алгоритм и исполнители» в базовом школьном курсе информатики».

Объект данной курсовой работы – это алгоритмическая содержательная линия школьного курса информатики, которая включает в себя вопросы, связанные с методами и средствами формализованного описания действий исполнителя.

Предмет – это методические особенности изучение раздела «алгоритм и исполнители» непосредственно в базовом курсе информатики.

Цель курсовой работы – раскрыть методические особенности изучения раздела «алгоритм и исполнители» в базовом школьном курсе информатики, представив разнообразные подходы, применяемые различными авторами для построения раздела, и выявив их существенные различия.

В соответствии с целью, предметом были определены следующие задачи исследования:

1. Провести теоретический анализ школьных учебников и литературных источников по теме исследования.
2. Уточнить основные понятия и категории.
3. Рассмотреть различные программы по теме исследования.
4. Построить план изучения раздела «Алгоритм и исполнители» в базовом курсе.

Методы исследования:

- теоретический анализ научной и учебно-методической литературы;

- методы презентации данных: таблицы.

В курсовой работе делается попытка анализа современных программ по разделу «Алгоритм и исполнители» и выявляется содержательная линии его преподавания.

**1. Теоретические основы раздела «Алгоритм и исполнители»**

**1.1 Определение основных понятий**

Изучение раздела «Алгоритмы и исполнители» обычно начинается с исторической справки.

Появление алгоритмов связано с именем математика Аль Хорезми, который сформулировал правила выполнения арифметических действий. Первоначально под алгоритмом понимали только правила выполнения четырех арифметических действий над числами. В дальнейшем это понятие стали использовать вообще для обозначения последовательности действий, приводящих к решению любой поставленной задачи. Само слово «алгоритм» возникло в Европе после перевода на латынь книги этого среднеазиатского математика, в котором его имя писалось как «Алгоритми».

Научное определение понятия алгоритма дал А.Черч в 1930 году. Позже и другие математики вносили свои уточнения в это определение.

Говоря об алгоритме вычислительного процесса, необходимо понимать, что объектами, к которым применялся алгоритм, являются данные. Алгоритм решения вычислительной задачи представляет собой совокупность правил преобразования исходных данных в результатные.

В дальнейшем дается определение понятие алгоритма.

Под ***алгоритмом*** понимают понятное и точное предписание (указание) исполнителю совершить последовательность действий, направленных на достижение указанной цели или на решение поставленной задачи.[1]

***Алгоритм*** *-* описание последовательности действий (план), исполнение которых приводит к решению поставленной задачи за конечное число шагов. [3]

***Алгоритм*** *-* понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.[9]

Всякий алгоритм составляется в расчете на определенного исполнителя. Им может быть человек, робот, компьютер и др. Вопрос о рассмотрении человека в этом качестве является спорным, но в рамках данной работы мы будем придерживаться мнения о человеке, как исполнителе алгоритма.

***Исполнитель***– объект, который выполняет алгоритм.[3]

Алгоритм может выполнить тот, кто понимает все его команды и может их выполнить. Таким образом, мы видим, что алгоритм не имеет смысла, если неизвестны или не учитываются возможности того, кто будет исполнять этот алгоритм, то есть возможности исполнителя. Поэтому нам потребуется еще одно определение «система команд исполнителя».

***Система команд исполнителя***– совокупность команд, которые данный исполнитель умеет выполнить*.* [3]

Многие алгоритмы, составленные для исполнителя-человека, часто предполагают наличие у человека некоторого дополнительного объема знаний, умений, интуиции и, естественно, здравого смысла. Например, в алгоритме перехода улицы предполагается, что исходное положение пешехода (исполнителя алгоритма) — лицом к улице, что он будет переходить улицу в разрешенном для этого месте. Предполагается также, что пешеход сообразит, что пропускать нужно транспорт, который не стоит, а движется, причем в сторону пешехода, и находится уже недалеко от перехода. Эти и множество других на первый взгляд мелочей нужно было бы обязательно учесть, если бы алгоритм предназначался для самостоятельных прогулок по городу робота.

Создание и использование в качестве исполнителей различных автоматов, роботов и компьютеров предъявляют очень строгие требования к точности описания алгоритмов их работы. Это связано с тем, что каждое автоматическое устройство имеет ограниченный, строго определенный набор законченных действий, которые (и только такие) оно может исполнять.

Для выполнения всякой работы, решения поставленной задачи исполнитель на входе получает алгоритм и исходные данные, а на выходе получает требуемые результаты.

Иногда при выполнении алгоритма возникает ситуация, когда исполнитель не может выполнить очередное предписание, несмотря на то что оно имеется в его системе команд. Такую ситуацию называют отказом.

Например, дан следующий алгоритм для исполнителя – человека.

1. Сделать один шаг вперед.

2. Повернуться направо.

3. Поднять вверх правую руку.

4. Поднять вверх левую руку.

5. Опустить вниз правую руку.

6. Поднять вверх левую руку

При выполнении шестой команды алгоритма исполнитель обнаруживает, что выполнить это предписание не удается, так как его рука уже находится в поднятом состоянии.

Как вы поняли, каждый алгоритм должен быть понятен исполнителю, поэтому алгоритм должен быть записан на понятном для исполнителя языке, и эта запись называется программой.

***Программа***- запись алгоритма на языке исполнителя.[9]

Основными ***свойствами*** алгоритма являются:

1. детерминированность (определенность). Это свойство указывает, что любое действие в алгоритме должно быть строго и недвусмысленно определено и описано для каждого случая;
2. результативность. Указывает на наличие таких исходных данных, для которых реализуемый по заданному алгоритму вычислительный процесс должен через конечное число шагов остановиться и выдать искомый результат;
3. массовость. Это свойство предполагает, что алгоритм должен быть пригоден для решения всех задач данного типа. Свойство массовости подразумевает использование переменных в качестве исходных данных алгоритма;
4. дискретность. Означает расчлененность определяемого алгоритмом вычислительного процесса на отдельные этапы, возможность выполнения которых исполнителем не вызывает сомнений. Только выполнив одну команду, исполнитель может приступить к выполнению следующей;
5. Понятность. Алгоритм должен быть понятен исполнителю и исполнитель должен быть в состоянии выполнить его команды.
   1. **Формы представления алгоритма**

Алгоритм должен быть формализован по некоторым правилам посредством конкретных изобразительных средств. К ним относятся следующие способы записи алгоритмов: словесный, формульно-словесный, графический, язык операторных схем, программа (алгоритмический язык).

Словесный способ представления несложен, но имеет недостатки. Главный недостаток состоит в том, что при таком способе допускается некоторая произвольность изложения, нет четких стандартов описания. Сложные задачи с анализом условий, с повторяющимися действиями и возвратами к предыдущим пунктам трудно представляются в словесном и словесно-формульном виде.

Наибольшее распространение благодаря своей наглядности получил графический способ записи алгоритмов. Одной из форм такого представления являются рисунки, но более строгая формализованная форма – это схемы или графы.

Наиболее распространенной формой представления алгоритма является блок-схема.

***Блок-схемой*** называется графическое изображение логической структуры алгоритма, в котором каждый этап процесса обработки информации представляется в виде геометрических символов (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых операций.

* ***Арифметический блок***

(операции присваивания)

Блок ввода – вывода

информации

* Условный (логический)

блок - проверка условия

* Блок начала – конца

алгоритма

* Соединитель – для соединения

удаленных блоков

Любой, даже самый сложный алгоритм, можно представить с помощью трех основных конструкций (структур): последовательности, ветвления и цикла. Каждая структура имеет один вход и один выход.

* В структуре «последовательность» действия выполняются последовательно, сверху вниз, без возвратов (рис. 1, а);
* В структуре «ветвление» выполняется либо одна, либо другая группа действий в зависимости от истинности (выполнения) или ложности (невыполнения) условия (рис. 1, б);
* В структуре «цикл» действия повторяются до тех пор, пока выполняется заданное условие (рис. 1, в).

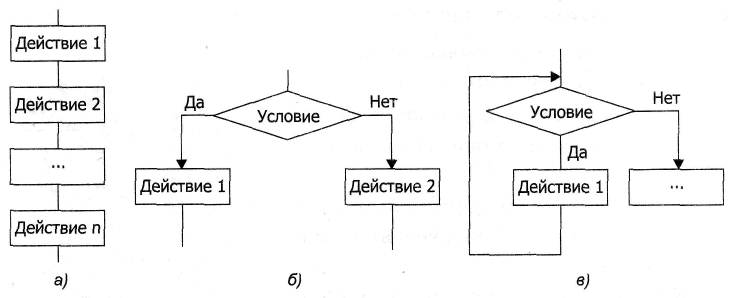


Рис. 1 – Выполнение заданных условий

В зависимости от того, какие базовые структуры использованы при составлении алгоритмов, различают три основные разновидности алгоритмов:

* линейный;
* ветвящийся;
* циклический.

***Линейным*** называется такой алгоритм, в котором все этапы решения задачи выполняются в естественном порядке следования записи этих этапов.

***Ветвящимся*** называется такой алгоритм, в котором выбор направления обработки информации зависит от исходных или промежуточных данных (от результатов проверки выполнения какого-либо логического условия).

Различают полную и неполную форму ветвления.

При полной форме ветвления действия выполняются в обоих случаях: и при истинности и при ложности условия. Ей соответствует следующее выражение: *если <условие>, то <действие 1>, иначе <действие 2>.*

Неполной форме ветвления соответствует выражение: *если <условие>, то <действие 1>.*

Циклом называется многократно повторяемый участок вычислений. Алгоритм, содержащий один или несколько циклов, называется ***циклическим***.

Основные понятия циклического алгоритма:

1. счетчик цикла – переменная, которая изменяет свое значение при переходе от цикла к циклу;
2. тело цикла – действия, которые повторяются;
3. начальное значение счетчика цикла – значение, от которого начинает изменяться счетчик цикла;
4. конечное значение счетчика цикла – значение, до которого изменяется счетчик цикла;
5. шаг – значение, на которое изменяется счетчик цикла.

По количеству выполнения циклы делятся на циклы с определенным (заранее заданным) числом повторений и циклы с неопределенным числом повторений. Количество повторений последних зависит от соблюдения некоторого условия, задающего необходимость выполнения цикла. При этом условие может проверяться в начале цикла — тогда речь идет о цикле с предусловием, или в конце — тогда это цикл с постусловием.

***Вспомогательный алгоритм*** — это блок последовательных действий в основном алгоритме, который выделен в качестве самостоятельного алгоритма, имеющего свое имя. [3]

Вспомогательные алгоритмы выступают в качестве сменных блоков алгоритма, которые могут быть составлены заранее и использованы в разных блок-схемах. Чем крупнее блоки, тем легче проходит сборка алгоритма. Вспомогательный алгоритм всегда является вложенным, если он включается в другой алгоритм. Но вложенная конструкция не является вспомогательным алгоритмом до тех пор, пока ей не дано имя.

К вспомогательным алгоритмам можно отнести процедуры, которые описываются перед выполнением основной программы и служат для выполнения одинаковых действий с различными параметрами.

При разработке алгоритма необходимо пройти минимум две стадии – сначала алгоритм должен быть понятен тому, кто его разрабатывает, а затем его следует преобразовать с учетом специфики среды. В том случае, если эти действия станет выполнять сам разработчик алгоритма, вторая стадия будет отсутствовать.

**1.3 Учебный алгоритмический язык**

Учебный алгоритмический язык - это средство для записи алгоритмов в виде, промежуточном между записью алгоритма на естественном (человеческом) языке и записью на языке ЭВМ (языке программирования).

К достоинствам учебного алгоритмического языка относится его простота, а также то, что алгоритм записывается на русском языке при помощи некоторого ограниченного числа слов, смысл и способ употребления которых строго определены. Эти слова называются служебными словами.

Для того чтобы выделять служебные слова среди других слов языка, их при письме подчеркивают.

Запись алгоритма на учебном алгоритмическом языке состоит из заголовка и тела алгоритма. Тело алгоритма заключается между ключевыми словами нач и кон и представляет собой последовательность команд алгоритма. Заголовок включает название алгоритма, отражающее его содержание, списки исходных данных (аргументов) и результатов.

Признаком заголовка алгоритма является ключевое слово алг.

Итак, алгоритм, записанный на учебном алгоритмическом языке, имеет следующую форму:

алг *название алгоритма*

aрг *список исходных данных*

рез *список результатов*

нач

*последовательность команд алгоритма*

кон

Изучение школьного алгоритмического языка целесообразно начать с команды присваивания, она является одной из основных команд.

Записывается она так:

<переменная> := <выражение>

Знак «: =» читается «присвоить».

В случае, когда величина, которой присваивается значение, входит и в правую часть команды, происходит следующее:

1) значение выражения, записанного в правой части команды присваивания, вычисляется с использованием текущих значений всех величин, входящих в это выражение;

2) переменной присваивается новое вычисленное текущее значение. При этом предшествующее значение переменной уничтожается.

Следовательно, команда b: = а + b означает, что к предыдущему текущему значению величины b прибавляется значение переменной а и полученный результат становится новым текущим значением величины b.

Этот пример иллюстрирует три основных свойства присваивания:

1) пока переменной не присвоено значение, она остается не определенной;

2) значение, присвоенное переменной, сохраняется в ней вплоть до выполнения следующего присваивания этой переменной нового значения;

3) новое значение, присвоенное переменной, заменяет ее предыдущее значение.

Теперь познакомимся с базовыми структурами, начнем с такой операции как "следование". Образуется последовательностью действий, следующих одно за другим:

действие 1

действие 2

. . . . . . . . .

действие n

Следующей рассмотрим базовую структуру "ветвление". Она обеспечивает в зависимости от результата проверки условия (да или нет) выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к общему выходу, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран. Структура ветвление существует в четырех основных вариантах:

1. если—то;

если *условие*

то *действия*

все

2. если—то—иначе;

если *условие*

то *действия 1*

иначе *действия 2*

все

3. выбор

выбор

при *условие 1: действия 1*

при *условие 2: действия 2*

. . . . . . . . . . . .

при *условие N: действия N*

{иначе *действия N+1*}

все

И, наконец, базовая структура цикл с помощью школьного алгоритмического языка будет выглядеть следующим образом.

***Цикл типа пока.***

Предписывает выполнять тело цикла до тех пор, пока выполняется условие, записанное после слова пока.

нц пока *условие*

*тело цикла*

(последовательность действий)

Кц

***Цикл типа для***

Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.

нц для i от i1 до i2

*тело цикла*

(последовательность действий)

кц

**1.4 Решение трудностей изучения раздела учащимися**

При изучении данной темы можно столкнуться со следующими сложностями:

* непонимание учащимися понятия «алгоритм»;
* неправильное приведение примеров алгоритма (не выполняются все его свойства, команды не входят в систему команд исполнителя);
* неправильное представление алгоритма в виде блок-схемы;
* неправильное использование простейших алгоритмических конструкций;
* отсутствие понимания принципа перевода алгоритма на формальный язык.

С первой из предложенного списка трудностей сталкиваются уже на начальном этапе изучения данной темы. В связи с этим раскрытие этого вопроса дается с использованием примеров, основанных на жизненном опыте учащихся. А также огромное значение имеет тот факт, что знакомство с алгоритмом происходит через раскрытие его неотъемлемых атрибутов или свойств, которые и позволяют некий текст (при условии представления его в словесной форме) воспринимать как алгоритм. Примеры, приводимые учителем во время урока должны отличаться разнообразием, касаться различных сфер человеческой деятельности. Это могут быть зарядка, выполняемая каждым человеком по утрам, переход через улицу, разведение костра, пришивание пуговицы, приготовление блюда по поваренной книги и другие.

Следующим этапом является закрепление понимания учащимися понятия алгоритм, и здесь возникает следующая проблема. Несмотря на то, что учащиеся знают определение и основные свойства они не могут правильно сформулировать примеры, которые являлись бы алгоритмами. Чаще всего забываются какие-нибудь важные атрибуты, им не уделяется достаточного внимания. Например, распространенной является ошибка, когда учащиеся забывают что необходимо, чтобы все действия приводили к какому-нибудь результату, а не выполнялись просто так.

Для закрепления основных понятий и для преодоления вышеназванных трудностей, связанных с определением алгоритма, полезно рассмотреть с учениками несколько заданий следующего содержания:

1. выполнить роль исполнителя: дан алгоритм, формально исполнить его;
2. определить исполнителя и систему команд для данного вида работы;
3. в рамках данной системы команд построить алгоритм;
4. определить необходимый набор исходных данных для решения задачи.

Следующая проблема связана с неправильным представлением алгоритма в виде блок-схемы. Чтобы избежать серьезных проблем с изучением этого подраздела, нужно выработать у учащихся практические навыки по разработке блок-схемы, для этого необходимо соблюдать единообразие представления основных алгоритмических конструкций, а также осуществлять многократное повторение упражнений на использованием данных схем.

Большую сложность у учащихся вызывает изучение базовых алгоритмических конструкций. Подробно необходимо остановить на каждой из них: линейность, ветвление и цикл.

Линейные алгоритмы воспринимаются легче всего, но необходимо подвести учащихся к выводу о невозможности их использования для большого круга задач.

При разборе конкретного алгоритма ветвления на схеме следует отметить разными цветами два возможных способа выполнения команды, точку входа и выхода из команды. Обязательно вслух проговаривается алгоритм с использованием ключевых слов «если», «то», «иначе». Это позволяет, во-первых, лучше усвоить данную структуру, а во-вторых, осуществить пропедевтику записи алгоритма с помощью псевдокода. Обращается внимание на то, что слева всегда записывается действие, которое будет выполнено в случае соблюдения условия, т. е. путь «да», а справа — действие, выполняемое при несоблюдении условия, т. е. путь «нет». Необходимо обратить внимание учащихся на то, что команда ветвления заканчивает свою работу, как только выполнится одна из двух предложенных команд.

Следует обсуждать с учащимися необходимость использования команды ветвления. Для этого можно задать следующие вопросы: почему алгоритм решения задачи не может иметь линейную структуру? Какое условие надо проверять при выполнении алгоритма? Какие действия выполняются при соблюдении условия, а какие — при его несоблюдении? Какая форма команды ветвления применена? В каких задачах используется данная структура алгоритма? и т. п.

В ряде учебников первой изучаемой конструкцией после команды следования является цикл, поскольку это дает возможность сократить запись алгоритма. Как правило, это конструкция «повторить n раз». Такой подход приводит к трудностям в освоении циклов как структуры организации действий, качественно отличающейся от линейной. Во-первых, другие разновидности цикла с предусловием и с постусловием (цикл «пока», цикл с параметром, цикл «до») воспринимаются как изолированные друг от друга и главный признак — повторяемость действий — не выступает в качестве системообразующего. Во-вторых, без внимания остаются опорные умения, которые необходимы при разработке циклов: правильное выделение условия продолжения или окончания цикла, правильное выделение тела цикла. Проверка условия в цикле «повторить п раз» практически не видна, и циклический алгоритм часто продолжает восприниматься учащимися как линейный, только иначе оформленный, что порождает неверный стереотип у учащихся в восприятии циклов вообще. Поэтому методически более целесообразным является изучение вначале команды ветвления, в которой используется условие, а уже затем команды повторения.

Изучение команды повторения следует начинать с введения цикла с постусловием, поскольку в этом случае учащемуся дается возможность вначале продумать команды, входящие в цикл, и только после этого сформулировать условие (вопрос) повторения этих команд. Если же сразу вводить цикл с предусловием, то учащимся придется выполнять оба эти действия одновременно, что снизит эффективность проведения занятий. В то же время цикл с постусловием рассматривается в качестве подготовки восприятия учащимися цикла с предусловием, обеспечивает перенос знаний на другой вид команды повторения, дает возможность работать по аналогии. Следует обратить внимание учащихся на то, что данные виды цикла отличаются по месту проверки условия, по условию возврата к повторению выполнения тела цикла. Если в команде повторения с постусловием тело цикла выполняется хотя бы один раз, то в команде повторения с предусловием оно может ни разу не выполняться.

Для решения последней трудности, связанной с отсутствие понимания принципа перевода алгоритма на формальный язык можно предложить следующее решение: необходимо начать изучение языка программирования с использования базовых алгоритмических конструкций, а также необходимо использовать параллельно различные формы представления алгоритма, что позволит более формально подойти к конкретному алгоритму.

**2. Методические особенности изучения раздела «Алгоритм и исполнители»**

**2.1 Содержание раздела в стандарте**

Прежде всего необходимо сказать, что общеобразовательный стандарт по информатике является нормативным документом, определяющим требования:

* к месту базового курса информатики в учебном плане школы;
* к содержанию базового курса информатики в виде обязательного минимума содержания образовательной области;
* к уровню подготовки учащихся в виде набора требований к знаниям, умениям, навыкам и научным представлениям школьников;
* к технологии и средствам проверки и оценки достижения учащимися требования образовательного стандарта.

Общеобразовательная область, представляемая в учебном плане школы курсом информатики, может быть рассмотрена в двух аспектах.

Первый аспект – системно-информационная картина мира, общие информационные закономерностям строения и функционирования самоуправляемых систем (биологические системы, общество, автоматизированные технические системы). Специфической особенностью этих систем является свойство их целесообразного функционирования, определяемое наличием в них органов, управляющих их поведением на основе получения, преобразования и целенаправленного использования информации.

Второй аспект данной общеобразовательной области–методы и средства получения, обработки, передачи, хранения и использования информации, решения задач с помощью компьютера и других средств новых информационных технологий. Этот аспект связан прежде всего с подготовкой учащихся к практической деятельности, продолжению образования.

Таким образом, совокупный предмет рассматриваемой общеобразовательной области имеет комплексный характер. Каждая его часть имеет различный удельный вес в реализации отдельных педагогических функций этой общеобразовательной области.

Анализ опыта преподавания курса основ информатики и вычислительной техники, новое понимание целей обучения информатике в школе, связанное с углублением представлений об общеобразовательном, мировоззренческом потенциале этого учебного предмета, показывает необходимость выделения нескольких этапов овладения основами информатики и формирования информационной культуры в процессе обучения в школе.

В данной курсовой работе рассматривается второй этап овладения основами информатики - базовый курс (VII–IX классы), обеспечивающий обязательный общеобразовательный минимум подготовки школьников по информатике. Он направлен на овладение учащимися методами и средствами информационной технологии решения задач, формирование навыков сознательного и рационального использования компьютера в своей учебной, а затем профессиональной деятельности. Изучение базового курса формирует представления об общности процессов получения, преобразования, передачи и хранения информации в живой природе, обществе, технике.

Представляется, что содержание базового курса может сочетать в себе все три существующих сейчас основных направления в обучении информатике в школе и отражающих важнейшие аспекты ее общеобразовательной значимости:

* мировоззренческий аспект, связанный с формированием представлений о системно-информационном подходе к анализу окружающего мира, о роли информации в управлении, специфике самоуправляемых систем, общих закономерностях информационных процессов в системах различной природы:
* “пользовательский” аспект, связанный с формированием компьютерной грамотности, подготовкой школьников к практической деятельности в условиях широкого использования информационных технологий;
* алгоритмический (программистский) аспект, связанный в настоящее время уже в большей мере с развитием мышления школьников.

Основные содержательные линии курса охватывают следующие группы вопросов:

* вопросы, связанные с пониманием сущности информационных процессов, информационными основами процессов управления в системах различной природы; вопросы, охватывающие представления о передаче информации, канале передачи информации, количестве информации (условно–“линия информационных процессов”);
* способы представления информации (условно–“линия представления информации”);
* методы и средства формализованного описания действий исполнителя (условно–“алгоритмическая линия”);
* вопросы, связанные с выбором исполнителя для решения задачи, анализом его свойств; возможностей и эффективности его применения для решения данной задачи (условно назовем эту линию “линией исполнителя”);
* вопросы, связанные с методом формализации, моделированием реальных объектов и явлений для их исследования с помощью ЭВМ, проведение компьютерного эксперимента (условно–“линия формализации и моделирования);
* этапы решения задач на ЭВМ, использование программного обеспечения разного типа для решения задач, представление о современных информационных технологиях, основанных на использовании компьютера (условна–“линия информационных технологий”).

Алгоритмическая линия включает в себя обязательный минимум содержания учебного материала, который должен быть усвоен учащимися полностью.

Изучение учебного материала данной содержательной линии курса обеспечивает учащимся возможность:

* понять (на основе анализа примеров) смысл понятия алгоритма, знать свойства алгоритмов, понять возможность автоматизации деятельности человека при исполнении алгоритмов;
* освоить основные алгоритмические конструкции (цикл, ветвление, процедура), применять алгоритмические конструкции для построения алгоритмов решения учебных задач;
* получить представление о “библиотеке алгоритмов”, уметь использовать библиотеку для построения более сложных алгоритмов;
* получить представление об одном из языков программирования (или учебном алгоритмическом языке), использовать этот язык для записи алгоритмов решения простых задач.

В образовательном стандарте также сформулированы основные требования к уровню подготовки учащихся.

Учащиеся должны:

* понимать сущность понимания алгоритма, знать его основные свойства, иллюстрировать их на конкретных примерах алгоритмов;
* понимать возможность автоматизации деятельности человека при исполнении алгоритмов;
* знать основные алгоритмические конструкции и уметь использовать их для построения алгоритмов;
* определять возможность применения исполнителя для решения конкретной задачи по системе его команд, построить и исполнить на компьютере алгоритм для учебного исполнителя (типа “черепахи”, “робота” и т.д.);
* записать на учебном алгоритмическом языке (или языке программирования) алгоритм решению простой задачи.

**2.2 Обзор авторских программ**

Понятие «алгоритм» является центральным в первом школьном учебнике под редакцией А.П. Ершова и В.М. Монахова – «Основы информатики и вычислительной техники». Указание на выполнение каждого отдельного действия названо командой, а «совокупность команд, которые могут быть выполнены исполнителем, называются системой команд исполнителя». В качестве основного свойства алгоритма подчеркивается формальный характер работы исполнителя при его выполнении. Отсюда делается вывод о том, что исполнителем может быть автомат (машина, робот). На этой идее основан принцип программного управления работой компьютера, поскольку программа – это и есть алгоритм, представленный на языке, «понятном» компьютеру – на языке программирования.

Сформированные в учебнике [1] понятия явились дидактической основой для раскрытия темы алгоритмизации во всех последующих учебниках информатики.

Практически весь алгоритмический раздел учебника ориентирован на исполнителя – человека. В задачах вычислительного характера (а их большинство в учебнике) в качестве метода работы исполнителя предлагается заполнение таблицы значений. В программировании такие таблицы принято называть трассировочными таблицами. В учебнике сказано: «При исполнении алгоритма компьютером значения величин хранятся в его памяти. При исполнении алгоритма человеком таблица значений выполняет роль дополнительной памяти для исполнителя».

Одним из основных методических достижений данного учебника стало введение в школьную информатику учебного алгоритмического языка. Алгоритмический язык А.П. Ершова можно назвать русскоязычным псевдокодом, предназначенным для обучения методике структурного программирования.

Наряду с использованием с использованием алгоритмического языка для описания алгоритмов в учебнике активно используются блок-схемы. Подчеркивается необходимость стандартного изображения блок-схем, чего также требует методика структурного подхода к программированию.

Теперь рассмотрим авторскую программу Макаровой Н.В. Раздел «алгоритм и исполнители» разбит на две темы, изучение которых происходит в рамках раздела «программное обеспечение информационных технологий». Программа рассчитана на преподавание информатики в расчете 2 часа в неделю.

Первая тема носит название «основы алгоритмизации» и включает в себя следующие основные вопросы: понятие и определение алгоритма; свойства алгоритмов; формы представления алгоритма: словесная, графическая, программа; типовые алгоритмические конструкции: последовательность, ветвление, цикл; стадии создания алгоритма; линейный алгоритм; разветвляющийся алгоритм; циклический алгоритм; цикл с известным числом повторений; цикл с предусловием; цикл с постусловием; вспомогательный алгоритм.

Вторая тема называется «представление о программе (классификация программ)», здесь рассмотрению подлежат следующие вопросы: исполнитель алгоритма; понятие программы и программирования; назначение процедуры; подходы к созданию программы: процедурный, объектный; классификация и характеристика программного обеспечения: системное, прикладное, инструменты программирования; роль программного обеспечения в организации работы компьютера.

Изучение раздела «алгоритм и исполнители» происходит на протяжении всего базового курса, то есть с 8 по 9 классы.

В 8 классе учащиеся знакомятся на примерах с понятием алгоритма и его основными свойствами. Учащиеся знакомятся с различными формами представления алгоритмов, останавливаются подробно на блок-схемах. Обучение происходит с параллельным освоением школьного алгоритмического языка. Таким образом, все типовые алгоритмические конструкции представлены одновременно с помощью блок-схем и Кумира (школьного алгоритмического языка), что позволяет обеспечить понимание формального представления алгоритма различными способами. На изучение данной темы отводится 6 часов занятий в некомпьютерном классе.

Что касается вопросов, связанных с исполнителем и системой его команд, то они рассматриваются уже в связи с формированием понятия программа и программирование. Здесь рассматриваются подробно различные подходы к созданию программы, а также большое внимание уделяется процедуре. На изучение данной темы отводится всего лишь 1 час занятий без использования компьютера.

А затем идет рассмотрение полностью практического вопроса, подводящему итог изучения теоретических основ построения алгоритмов и программ, «среда программирования». В данном учебнике для обучения учащихся алгоритмизации предлагается язык программирования ЛОГО. В состав данного языка входит исполнитель Черепашка, назначение которого -изображение на экране чертежей, рисунков, состоящих из прямолинейных отрезков. Программы управления Черепашкой составляются из команд: вперед(а), назад(а), направо(в), налево(в), поднять хвост, опустить хвост. Имеется в виду, что черепашка рисует хвостом, и если хвост опущен, то при перемещении проводится линия, а когда хвост поднят, то линия не рисуется. Кроме того в языке имеются все основные структурные команды. В целом ЛОГО предназначен для обучения структурной методики программирования.

Главное методическое достоинство Черепашки – ясность для ученика решаемых задач, наглядность процесса работы в ходе выполняемой программы. А как известно дидактический принцип наглядности является одним из важнейших в процессе обучения. Всего на изучение этой темы отводится 8 часов, что не является достаточным для разбора всех тонкостей этой среды, но хватает для знакомства с одной из сред программирования.

В 9 классе происходит более подробное изучение темы «основы алгоритмизации», на этот раздел выделяется уже 8 часов. В этом классе происходит повторение всех изученных понятий и их свойств с более глубоким проникновением в тему благодаря владению учащимися различными формами представления алгоритмов: блок-схемы, школьный алгоритмический язык, среда программирования ЛОГО. Учащиеся более подробно останавливаются на рассмотрении циклических алгоритмов, в частности они изучают различные разновидности циклических алгоритмов: цикл с предусловием, цикл с постусловием, цикл с известным числом повторений. Также они начинают знакомиться с новой средой программирования Visual Basic, хотя это происходит с помощью рассмотрения аналогов алгоритмов, записанных на изученных раннее алгоритмических языках (ЛОГО, Кумир).

Следующей авторской программой, которая будет рассмотрена в рамках данной курсовой работы является программа Угриновича Н.

Данная учебная программа составлена с расчетом на 1 урок информатики в неделю. Рассмотрение раздела «алгоритм и исполнители» начинается с 9 класса и входит как отдельная тема в главу «основы алгоритмизации и объектно-ориентированного программирования». На изучение данной темы отводится 7 часов.

Изучение начинается с рассмотрения вопросов, связанных с алгоритмом и его формальным представлением, при этом определение алгоритма дается только после объявления и пояснения примерами его основных свойств. Здесь не так много приводится примеров алгоритмов по сравнению с учебником Макаровой Н.В., что усложняет формирование у учащихся библиотеки алгоритмов. Достаточно подробно рассматривается понятие исполнитель алгоритма, в качестве которого упор делается на компьютер и человека. Все последующие уроке посвящены непосредственно изучение объектно-ориентированного программирования на примере языка Visual Basic. Также параллельно с этим происходит преставление алгоритмов с помощью редактора блок-схем алгоритмов Block-diagram editor.

С одной стороны система объектно-ориентированного визуального программирования Visual Basic является системой программирования, так как позволяет кодировать алгоритмы на этом языке. С другой стороны, она является средой проектирования, так как позволяет осуществлять визуальное конструирование графического интерфейса. В данной теме также рассматривается понятие проекта, который является результатом процессов программирования и проектирования.

В рамках системы программирования Visual Basic учащиеся изучают основные алгоритмические структуры (линейная, ветвление, выбор, цикл), типы, имена и значения переменных, операцию присваивания, а также графические возможности языка программирования.

В качестве формы отчетности учащимися выполняются различные проекты, соответствующие теме урока.

Следующая авторская программа, которой необходимо уделить внимание является программа по учебному курсу «информатика и ИКТ» является программа Семакина И.Г. Она предусматривает изучение данного курса на базовом уровне в расчете – 2 урока в неделю.

В 9 классе тема «алгоритм и исполнители» входит в раздел «информация и управление», таким образом можно отметить несколько иной подход к пониманию алгоритмизации. Его можно назвать кибернетическим подходом. Алгоритм трактуется как информационный компонент системы управления. Такой подход дает возможность ввести в содержание базового курса новую содержательную линию – линию управления. Это многоплановая линия, которая позволяет затронуть следующие вопросы:

* элементы теоретической кибернетики: кибернетическая модель управления с обратной связью;
* элементы прикладной кибернетики: структура компьютерных систем автоматического управления (системы с программным управлением); назначение автоматизированных систем управления;
* основы теории алгоритмов.

На изучение этого раздела отводится 10 часов, в том числе 5 часов теории и столько же практики.

Основные теоретические вопросы, которые следуют обязательному рассмотрению: управление и кибернетика, автоматизированные и автоматические системы управления; определение и свойства алгоритма; линейные алгоритмы; вспомогательные алгоритмы и подпрограммы; циклические алгоритмы и ветвления.

Практическая работа осуществляется с помощью одного из учебных графических исполнителей алгоритмов: Черепашки, Кенгуренка или Чертежника.

Исполнитель Чертежник – это своеобразный графопостроитель, действующий в системе декартовых координат, связанных с экраном. Его назначение – изображение чертежей, графиков, рисунков, состоящих из прямолинейных отрезков. Чертежник близок к Черепашке, однако ее работа не связана с системой координат (хотя единицы длины для нее существуют).

Кенгуренок является аналогом исполнителя Чертежник. Кенгуренок может работать в режиме прямого управления: «команда - исполнение», и в режиме программного управления: «программирование – автоматическое исполнение программы».

**2.3 Методические особенности изучения раздела**

Проанализировав авторские программы, используемые в настоящее время в различных учебных заведениях и одобренные Министерством образования и науки Российской Федерации, можно сделать вывод о различны подходах к построению изучения раздела «алгоритм и исполнители».

В отношении теоретического материала можно сказать, что существенных отличий в различных программах не видно, но значительно различается порядок изучения отдельных подразделов. Например, в некоторых учебниках понятия алгоритм дается в непосредственной связи с понятием исполнителя, а в других эти темы достаточно разделены и предусматривают изучение исполнителя на примере конкретной среды программирования.

Но существенные различия начинаются при рассмотрении практической составляющей данного раздела, хотя они разрешены стандартом. Предусматривается выбор различных сред программирования авторами программ в соответствии с их концепцией построения учебного материала. Но для всех программ характерно изучение школьного алгоритмического языка, как основы для понимания простейших алгоритмических конструкций и основных этапов построения программ.

Поэтому следует более подробно остановиться на вопросах, связанных с различными языками программирования.

Поскольку в базовом курсе ставится лишь цель первоначального знакомства с программированием, то строгого описания языка программирования не требуется. Основной используемый метод – демонстрация языка на примерах простых программ с краткими комментариями. В связи с этим у учащихся могут возникнуть трудности с изучением объектно-ориентированной среды Visual Basic. Некоторые понятия достаточно воспринять ученикам на «интуитивном» уровне. Для выполнения учениками несложных самостоятельных заданий достаточно действовать методом «по образцу».

Учитель может задуматься над проблемой: как лучше связать изучение методов построения алгоритмов работы с величинами и языка программирования. Здесь возможны два варианта:

1. сначала рассматриваются всевозможные алгоритмы, для описания которых используются блок-схемы и алгоритмический язык, а затем – правила языка программирования, способы перевода уже построенных алгоритмов программу на этом языке;
2. алгоритмизация и языки программирования осваиваются параллельно.

Теоретическое изучение алгоритмизации и программирования, оторванное от практики, малоэффективно. Желательно, чтобы ученики как можно раньше получили возможность проверять правильность своих алгоритмов, работая на компьютере.

Обучение программированию должно проводиться на примере типовых задач с постепенным усложнением структуры алгоритмов.

В качестве примера построение последовательности изучения темы «алгоритм и исполнители» можно предложить следующее поурочное планирование, разработанное по учебнику Семакина (9 класс) [9].

Таблица 2. – Основные темы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Тема | Всего часов | теория | практика | Вид практического занятия | Программное обеспечение |
|  | Управление и кибернетика. Автоматизированные и автоматические системы управления. | 1 | 1 | - |  |  |
|  | Определение и свойства алгоритма. Линейные алгоритмы. | 1 | 1 | - |  |  |
|  | ***Практическая работа №23.*** Разработка линейных алгоритмов для графического исполнителя. | 1 | - | 1 | Практикум на ПК | Графический исполнитель Черепашка, Кенгуренок, Чертежник |
|  | Вспомогательные алгоритмы и подпрограммы. Последовательная детализация. | 1 | 1 | - |  |  |
|  | ***Практическая работа №23.*** Учебный исполнитель алгоритмов. Последовательная детализация. | 1 | - | 1 | Практикум на ПК | Графический исполнитель Черепашка, Кенгуренок, Чертежник |
|  | Циклические алгоритмы. | 1 | 1 | - |  |  |
|  | ***Практическая работа №24.*** Учебный исполнитель алгоритмов. Циклические алгоритмы. | 1 | - | 1 | Практикум на ПК | Графический исполнитель Черепашка, Кенгуренок, Чертежник |
|  | Ветвления. Циклы в сочетании с ветвлениями. | 1 | 1 | - |  |  |
|  | ***Практическая работа №25.*** Учебный исполнитель алгоритмов. Ветвления. Циклы в сочетании с ветвлениями. | 1 | - | 1 | Практикум на ПК | Графический исполнитель Черепашка, Кенгуренок, Чертежник |
|  | ***Контрольная работа №6.*** Информация и управление (тестирование, зачетная практическая работа, защита проекта). | 1 | - | 1 |  |  |
|  | ***Итого*** | ***10*** | ***5*** | ***5*** |  |  |

Согласно с данным поурочным планированием на изучение раздела «алгоритм и исполнители» в 9 классе по программе И.Г. Семакина отводится 10 учебных часов, из которых 5 посвящены теории и 5 практическому освоению учебной среды программирования. Учащиеся на примерах знакомятся с базовыми алгоритмическими конструкциями, тем самым на данном этапе закладываются первые навыки, необходимые для дальнейшего совершенствования в области освоения более сложных сред программирования.

В конце освоения раздела предусмотрен контроль знаний, включающий в себе проведение контрольной работе в форме, соответствующей уровню подготовки учащихся и оптимальностью использования того или иного метода контроля.

Для сравнения нужно рассмотреть поурочное планирование, составленное на основе другой авторской программы – это программа Н.Д. Угриновича. По ее основе составлено поурочное планирование предмета «Информатика и ИКТ», а именно раздела «алгоритм и исполнители» для 9 класса базового уровня.[12]

Таблица 3. – Темы и программное обеспечение

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Тема | Кол-во часов | Тип урока | Контроль | Программное обеспечение |
| 1 | Понятие алгоритма, свойства алгоритмов. | 1 | комбинированный |  |  |
| 2 | Исполнители алгоритмов, система команд исполнителя. Способы записей алгоритмов. Формальное исполнение алгоритмов. | 1 | комбинированный |  | Редактор блок-схем алгоритмов Block-diagram editor  Система объектно-ориентированного программирования Visual Basic 2005 Express Edition |
| 3 | Объектно-ориентированное программирование. Графический интерфейс: форма и управляющие элементы. Событийные процедуры. | 1 | комбинированный |  | Редактор блок-схем алгоритмов Block-diagram editor  Система объектно-ориентированного программирования Visual Basic 2005 Express Edition |
| 4 | Тип, имя и значение переменной. Присваивание. | 1 | комбинированный | № 33. Проект «Форма и размещение на ней управляющих элементов». | Редактор блок-схем алгоритмов Block-diagram editor  Система объектно-ориентированного программирования Visual Basic 2005 Express Edition |
| 5 | Основные алгоритмические структуры (линейная, ветвление, выбор, цикл) и их кодирование на языке программирования. | 1 | комбинированный | № 34. Проект «Линейный алгоритм. Тип, имя и значение переменных».  № 35. Проект «Ветвление. Проверка знаний».  № 36. Проект «Выбор. Выставление оценки». | Редактор блок-схем алгоритмов Block-diagram editor  Система объектно-ориентированного программирования Visual Basic 2005 Express Edition |
| 6 | Графические возможности языка программирования. | 1 | комбинированный | № 37. Проект «Цикл. Коды символов».  № 38. Проект «Графический редактор». | Редактор блок-схем алгоритмов Block-diagram editor  Система объектно-ориентированного программирования Visual Basic 2005 Express Edition |
| 7 | Контрольная работа: тестирование, разработка зачетного проекта. | 1 | контроль знаний |  | Редактор блок-схем алгоритмов Block-diagram editor  Система объектно-ориентированного программирования Visual Basic 2005 Express Edition |

Первое отличие поурочного планирования по учебнику Н.Д. Угриновича от планирования по программе И.Г. Семакина заключается в количестве часов, отводимое на изучение этого довольно объемного и сложного для понимания раздела школьной программы – их всего 7, включая и контроль знаний. Этого явно недостаточно для глубокого и детального изучения материала, но вполне хватает для поверхностного знакомства с алгоритмами и их конкретном представлении на языке программирования.

Второе существенное отличие данной программы, как было отмечено раньше, это использование с самого начала обучения довольно сложной среды объектно-ориентированного программирования - Visual Basic 2005, что также вызывает дополнительные сложности у учащихся. Также параллельно с Visual Basic 2005 на уроках используется редактор блок-схем алгоритмов Block-diagram editor, что обеспечивает многосторонний подход для осмысления учащимися понятия структуры алгоритма, что в свою очередь обеспечивает лучшее понимания этапов работы программы, тем самым помогает понять логические основы программирования.

В конце раздела также предусматривается контрольная работа, в форме наиболее оптимальной для учащихся.

**Заключение**

В ходе проведенного исследования был осуществлен теоретический анализ учебников и литературных источников по теме, который позволяет сделать следующие выводы.

Раздел «Алгоритм и исполнители» в различных учебных пособиях освещен по разному, но несмотря на это в данной курсовой работе удалось представить необходимый минимум учебного материала, который подлежит обязательному рассмотрению. В дальнейшем приводится подтверждение целесообразности такого выбора в соответствии с образовательным стандартом и требованиями к знаниям учащихся, заключенными в нем. В связи с этим все рассмотренные учебники раскрывают вопросы, связанные с рассмотрением понятий – алгоритм, свойства алгоритма, исполнитель, алгоритма, система команд исполнителя, формы представления алгоритма, базовые алгоритмические конструкции, программа, среда программирования и т.д. Важным является на данном этапе рассмотрение различных исполнителей алгоритма, в том числе человека и компьютера.

На основе анализа различной литературы доказывается необходимость рассмотрения такой формы представления алгоритма как блок-схема, так как она помогает обеспечить структурное представление алгоритма, что играет немаловажную роль при формировании алгоритмического мышления.

Также можно сделать вывод, что теоретическое изучение алгоритмизации и программирования малоэффективен. Поэтому возникает необходимость преставления алгоритмов с помощью специально разработанного языка; на ранних этапах для подкрепления теоретических сведений наиболее оптимальным является изучение школьного алгоритмического языка.

На основании рассмотрения образовательного стандарта по информатике можно сделать вывод о том, что алгоритмическая содержательная линия является одной из ключевых в изучении данного предмета и тесно связана со всеми остальными образовательными направлениями, а особенно с линией исполнителя (компьютера).

Также в ходе проведения исследования был проведен анализ авторских программ, а конкретно раздела «алгоритм и исполнители», который позволяет сделать следующие выводы:

- содержание теоретического составляющего практически полностью совпадает;

- практическая составляющая различается в следствие выбора авторами различных сред реализации программ (ЛОГО, Черепашка, Visual Basic);

- различается время первого знакомство учащихся с основами алгоритмизации (с 8 или 9 класса);

- различается количество учебных часов, отводимых различными авторами на изучении данного раздела.

На основе всего вышесказанного были предложены несколько концепций преподавания разделам «алгоритм и исполнитель» в соответствии с различными авторскими программами. А также был проведем анализ основных трудностей, с которыми сталкиваются учащиеся и предложены варианты их решения.

Также в ходе исследования была решена задача уточнения основных понятий и категорий. В курсовой работе были рассмотрены и охарактеризованы такие понятия, как алгоритм, свойства алгоритма, исполнитель, алгоритма, система команд исполнителя.

Таким образом, в ходе проведенного исследования нами были решены все поставленные задачи.

Таким образом, можно сделать вывод о необходимости практического подкрепления данного раздела, основанного на различных формах построения алгоритма, но при этом основной упор должен быть сделан на одну конкретную среду программирования для формирования навыков построения целой библиотеки алгоритмов с возможностью дальнейшего освоения других сред.

**Библиографический список**

1. Изучение информатики и вычислительной техники: Пособие для учителей / Под ред. А.П. Ершова, В.М. Монахова. – М., 2003.
2. Информатика и ИКТ. Практикум 8-9 класс / Под ред. Н.В. Макаровой– СПб, 2008.
3. Информатика и ИКТ. Учебник 8-9 класс / Под ред. Н.В. Макаровой– СПб, 2008.
4. Лапчик М.П. Вычисление. Алгоритмизация. Программирование: Пособие для учителя / Лапчик М.П. – М., 2002.
5. Лапчик М.П. Методика обучения информатики / Лапчик М.П. – М., 2001.
6. Пейперт С. Дети, компьютер и плодотворные идеи. / Пейперт С. – М, 2003.
7. Программы для средних общеобразовательных учебных заведений. Информатика и ИКТ. / М., 2008.
8. Программа по информатике и ИКТ. 5-11 класс / Под ред. Н.В. Макаровой – СПб, 2008.
9. Семакин И.Г. Базовый курс информатики: 7-9 класс / Семакин И.Г и др. – М., 2008.
10. Семакин И.Г., Шеина Т.Ю. Преподавание базового курса информатики в средней школе: Метод. пособие / Семакин И.Г., Шеина Т.Ю. – М., 2002.
11. Сластенин В.А. и др. Педагогика: Учеб. пособие для студентов пед. Учебных заведений / Сластенин В.А. и др., М. – 1999.
12. Угринович Н. Информатика. Базовый курс: Учебник для 9 класса / Угринович Н.Д. – М., 2008.
13. Угринович Н. Преподавание курса «Информатика и ИКТ» в основной и старей школе. Методическое пособие для учителей/ Угринович Н.Д. – М., 2008.