Апольских Евгения Ивановна,
старший преподаватель кафедры
теоретических основ информатики

Подходы к изучению понятия «алгоритм», «исполнитель» и др. Методика введения понятии алгоритма. Методика обучения алгоритмизации на учебных исполнителях.

Для реализации нового ФГОС были разработаны примерные рабочие программы по учебным предметам. Примерная рабочая программа определяет количественные и качественные характеристики учебного материала для каждого года изучения.

Основной линией обучения в школьном курсе информатики является линия «Алгоритмизация и программирование». Изучение алгоритмизации в школьном курсе информатике может иметь два целевых аспекта: первый (на базовом уровне) – развивающий аспект, под которым понимают развитие алгоритмического мышления учащихся; второй (на углубленном уровне) – программистский аспект, под которым понимают развитие навыков составление программ на языках программирования.

Рассмотрим содержанием раздела «Алгоритмизация и программирование» в примерных рабочих программах основного общего образования.

На базовом уровне в 7 классе раздел «Алгоритмизация и программирование» не рассматривается.

На углубленном же на уровне на данный раздел отводится 24 часа, который разбивается на два подраздела:

- Исполнители и алгоритмы. Алгоритмические конструкции (16 часов).
- Компьютерная графика и анимация (8 часов) (графические возможности языков программирования).

В 8 классе на «Алгоритмизацию и программирование» на базовом уровне отводится 42 часа и рассматриваются следующие темы:

- Алгоритмы и программирование (21 час).
- Исполнители и алгоритмы. Алгоритмические конструкции (10 часов).
- Язык программирования (9 часов).
- Анализ алгоритмов (2 часа).

На углубленном уровне в 8 класс на данную линию отводится 68 часов и рассматриваются темы:

- Алгоритмы и программирование (34 часа).
- Язык программирования (34 часа).

В 9 классе на «Алгоритмизацию и программирование» на базовом уровне отводится 16 часов и рассматриваются следующие темы:

- Алгоритмы и программирование (8 часов).
- Разработка алгоритмов и программ (6 часов).
- Управление (2 часа).

На углубленном уровне отводится 56 часов и рассматриваются следующие темы:

- Алгоритмы и программирование (28 часов).
- Разработка алгоритмов и программ (24 часа).
- Управление (4 часа).

Изучение алгоритмизации начинается с введения понятия алгоритма. Понятие алгоритма – одно из фундаментальных понятий информатики. Алгоритмизация наряду с моделированием выступает в качестве общего метода информатики.

Особенность положения состоит в том, что при решении практических задач, предполагающих разработку алгоритмов для реализации на ЭВМ, и тем более при использовании на практике информационных технологий, можно, как правило, не опираться на высокую формализацию данного понятия. Поэтому представляется целесообразным познакомиться с алгоритмами и алгоритмизацией на основе содержательного толкования сущности понятия алгоритма и рассмотрения основных его свойств. При таком подходе алгоритмизация более выступает как набор определенных практических приемов, особых специфических навыков рационального мышления в рамках заданных языковых средств.

Понятие алгоритма относится к исходным математическим понятиям, поэтому не может быть определено через другие, более простые понятия. Из-за этого определение алгоритма в школьных учебниках по информатике отличается большим разнообразием.

В учебнике И.Г. Семакина и др. алгоритм определяется как последовательность команд, управляющих работой какого либо объекта, и далее дается более строгое определение – понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.

В учебнике Л.Л. Босовой понятие алгоритма вводится как описание последовательности шагов в решении задачи, приводящих от исходных данных к требуемому результату.

В учебнике К.Ю. Полякова алгоритмом называют точный набор инструкций для исполнителя, который приводит к решению задачи за конечное время.

Определения понятия «алгоритм», которые мы с вами рассмотрели, часто называют *интуитивными,* потому что они содержат такие «нематематические» понятия как «точный набор», «инструкция», «исполнитель», «решение задачи». Эти термины невозможно записать строго, используя язык математики и логики, поэтому для математического доказательства приведенные определения не подходят.

Понятие алгоритма полнее раскрывается через его свойства, которые обеспечивают его автоматическое исполнение. Основных свойств алгоритма – пять. Рассмотрим их:

Дискретность. Это свойство указывает на то, что любой алгоритм должен состоять из законченного числа шагов, где шаг – это элементарное действие, которому присвоен номер. Нумерация шагов производится сверху вниз и слева направо.

Детерминированность (определённость). Это однозначная трактовка содержимого каждого шага алгоритма и точное их исполнение.

Результативность. Алгоритм должен приводить к решению поставленной задачи за конечное число шагов.

Массовость. Алгоритм решения задачи должен разрабатываться не для одной конкретной задачи, а для целого класса однотипных задач, различающихся только исходными данными.

Понятность. Содержание шагов алгоритма должно быть понятно исполнителю, то есть инструкции алгоритма должны входить в систему команд исполнителя.

При рассмотрении свойства понятность, учителю следует акцентировать внимание учащихся на том, что алгоритм всегда составляется с ориентацией на исполнителя алгоритма.

Исполнитель – это объект или субъект, для управления которым составляется алгоритм. В этом случае учителю следует привести примеры алгоритмов для управления действиями различных субъектов (исполнителей).

Основной характеристикой исполнителя алгоритма является система команд исполнителя (СКИ), которая определяется как конечное множество команд (элементарных действий), которые понимает исполнитель и способен их выполнять. В этом месте учителю следует привести пример какой-либо системы команд. Далее следует остановиться на том, что алгоритм может включать в себя только те команды, которые входят в его СКИ. Алгоритм не должен быть рассчитан на принятие исполнителем самостоятельных решений, не предусмотренных составителем алгоритма.

Одно из принципиальных обстоятельств состоит в том, что исполнитель не вникает в смысл того, что он делает, но получает необходимый результат. В таком случае говорят, что исполнитель действует формально, т.е. отвлекается от содержания поставленной задачи и только строго выполняет некоторые правила, инструкции.

Это – важная особенность алгоритмов. Наличие алгоритма формализует процесс решения задачи, исключает рассуждение исполнителя. Использование алгоритма дает возможность решать задачу формально, механически исполняя команды алгоритма в указанной последовательности.

Главной целью раздела алгоритмизации является овладение учащимися структурной методикой построения алгоритмов. Традиционно применяемым дидактическим средством в этом разделе являются учебные исполнители алгоритмов.

Цель обучения алгоритмизации заключается в овладении учащимися методикой построения алгоритмов. Это значит, ученики должны научиться

использовать на практике основные управляющие структуры: следование, ветвление, цикл; уметь разбивать задачу на подзадачи, применять метод последовательной детализации алгоритма.

Учебный материал необходимо подбирать так, чтобы:

- задачи шли идти от простого к сложному;

- в каждой задаче была новизна;

- использовалась наследование – каждая последующая задача решается, используя знания, полученных при решении предыдущих задач.

В качестве дидактических средств удобно использовать учебные исполнители алгоритмов: Кузнечик, Чертежник, LOGO, КуМир, Scratch и т.д.

Учебные исполнители алгоритмов – это программные средства, предназначенные для обучения алгоритмизации.

Использование исполнителей с методической точки зрения очень эффективно. Основные достоинства – наглядность работы исполнителя, понятность решаемых задач и повышает интерес к процессу решения задачи. Как известно, дидактический принцип наглядности является одним из важнейших в процессе любого обучения.

Учебный исполнитель должен удовлетворять условиям:

1. Исполнитель должен работать «в обстановке».
2. Исполнитель должен имитировать процесс управления некоторым реальным объектом, например роботом, черепахой, чертежником и др.
3. В системе команд исполнителя должны быть представлены все основные структурные команды управления – циклы, ветвления.
4. Исполнитель должен позволять использовать вспомогательные алгоритмы (процедуры).

В примерных программах основного общего образования в качестве основного учебного исполнителя указывается система программирования КуМир, в состав которой входят такие исполнители как Робот, Черепашка, Чертёжник.

Язык КуМир – универсальный язык программирования, его прототипом послужил «школьный язык программирования», разработанный А.П. Ершовым в первой половине 80-х годов XX века.

Особенности системы КуМир

- В системе КуМир используется школьный алгоритмический язык с русской лексикой и встроенными исполнителями Робот и Чертёжник.
- При вводе программы КуМир осуществляет постоянный полный контроль ее правильности, сообщая на полях программы обо всех обнаруженных ошибках.
- При выполнении программы в пошаговом режиме КуМир выводит на поля результаты операций присваивания и значения логических выражений. Это позволяет ускорить процесс освоения азов программирования.
- КуМир работает в операционных системах Windows, MacOS и GNU/Linux.

Система КуМир позволяет создавать, отлаживать и выполнять программы. Несложные программы Вы сможете начать писать и выполнять практически сразу после знакомства с системой, однако система КУМИР позволяет создавать и достаточно большие, сложные программы. Во время редактирования программы система КУМИР автоматически производит синтаксический разбор и сообщает о найденных ошибках.

В то же время КУМИР является учебной системой. В неё встроено несколько графических исполнителей, действия которых визуализируются на их «игровом поле» или арене.

Язык КуМир ни в коем случае не должен стать конечной точкой при обучении программированию! Это язык, с которого хорошо начать, чтобы освоить основы алгоритмического подхода и процедурный стиль программирования. Следующими языками должны стать современные языке программирования общего назначения (Python, C++, Java, C#).