МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ "ОСНОВНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА № 34" Г. СЫКТЫВКАРА
Подписано цифровой подписью: МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ "ОСНОВНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА № 34" Г. СЫКТЫВКАРА
Дата: 2023.10.06 08:43:07 +03'00'

МИНИСТЕРСТВОПРОСВЕЩЕНИЯРОССИЙСКОЙФЕДЕРАЦИИ

МинистерствообразованияинаукиимолодежнойполитикиРеспубликиКоми

УправлениеобразованияМОГО"Сыктывкар"

МОУ"ООШ№34"г.Сыктывкара

РАССМОТРЕНО
Педагогическим советом
Протокол № 13
от «30» августа 2023 г.

УТВЕРЖДЕНО
Приказом № 360
от «30» августа 2023 г.

РАБОЧАЯ ПРОГРАММА

учебного модуля

«Практикум по основам алгоритмизации и программирования»

для 8 класса основного общего образования

на 2023-2024 учебный год

Составитель:Покровкова Людмила Александровна

учительинформатики

Сыктывкар 2023

Содержание

1. Пояснительная записка

Рабочая программа по учебному модулю «Практикум по основам алгоритмизации и программирования»разработана для обучения учащихся **8** классов МОУ «Основная общеобразовательная школа № 34» г. Сыктывкара» (далее - МОУ «ООШ № 34» г. Сыктывкара) **в соответствии с:**

- Приказом Минпросвещения России от 18.05.2023 N 370 «Об утверждении федеральной образовательной программы основного общего образования» (Зарегистрировано в Минюсте России 12.07.2023 N 74223)
- Федеральным государственным образовательным стандартом основного общего образования, утверждённым приказом Министерства образования и науки Российской Федерации от 17 декабря 2010 г № 1897 в действующей редакции;
- Приказом Минпросвещения России от 11.12.2020 г. №712 «О внесении изменений в некоторые федеральные государственные образовательные стандарты общего образования по вопросам воспитания обучающихся».

Наоснове:

- Требований к результатам освоения основной образовательной программы основного общего образования МОУ «ООШ № 34» г. Сыктывкара;
- Методических рекомендаций МУ ДПО «ЦРО» по доработке рабочих программ учебных предметов в связи с рабочей программой воспитания.
- Положения о рабочей программе учебного предмета, утвержденного приказом МОУ «ООШ №34» г. Сыктывкара
- УМК БосовойЛ.Л., Босовой А.Ю. «Информатика».

С учетом:

- Примерной основной образовательной программы основного общего образования (протокол № 1/20 от 04.02.2020);
- Авторской программы: Информатика. Программа для основной школы: 5-6 классы. 7-9 классы / Л, Л. Босова, А. Ю. Босова. - М. : БИНОМ. Лаборатория знаний, 2014.

При реализации РПУП побуждение обучающихся соблюдать на уроке общепринятые нормы поведения, правила общения осуществляется посредством следования правилам, вытекающих из ценностей школы, выработка и принятие которых описаны в РПВ (модуль «Школьный урок»).

Программа по учебному модулю «Практикум по основам алгоритмизации и программирования»для основной школы составлена в соответствии с: требованиями Федерального государственного образовательного стандарта основного общего образования

(ФГОС ООО); требованиями к результатам освоения основной образовательной программы (личностным, метапредметным, предметным); основными подходами к развитию и формированию универсальных учебных действий (УУД) для основного общего образования. В ней соблюдается преемственность с федеральным государственным образовательным стандартом начального общего образования; учитываются возрастные и психологические особенности школьников, обучающихся на ступени основного общего образования, учитываются межпредметные связи.

Методологической основой федеральных государственных образовательных стандартов является системно-деятельностный подход, в рамках которого реализуются современные стратегии обучения, предполагающие использование информационных и коммуникационных технологий (ИКТ) в процессе изучения всех предметов, во внеурочной и внешкольной деятельности на протяжении всего периода обучения в школе. Организация учебно-воспитательного процесса в современной информационно-образовательной среде является необходимым условием формирования информационной культуры современного школьника, достижения им ряда образовательных результатов, прямо связанных с необходимостью использования информационных и коммуникационных технологий.

Средства ИКТ не только обеспечивают образование с использованием той же технологии, которую учащиеся применяют для связи и развлечений вне школы (что важно само по себе с точки зрения социализации учащихся в современном информационном обществе), но и создают условия для индивидуализации учебного процесса, повышения его эффективности и результативности. На протяжении всего периода существования школьного курса информатики преподавание этого предмета было тесно связано с информатизацией школьного образования: именно в рамках курса информатики школьники знакомились с теоретическими основами информационных технологий, овладевали практическими навыками использования средств ИКТ, которые потенциально могли применять при изучении других школьных предметов и в повседневной жизни.

Изучение алгоритмов и языков программирования является неотъемлемой частью информатики. Программирование вырабатывает у учащихся логическое, комбинаторное и алгоритмическое мышление, творческие способности. Обучение программированию – наиболее разработанная часть методики обучения информатике, ибо имеет давнюю историю. Первые годы введения в школе информатики как обязательного предмета школьники изучали, в основном, именно программирование. Основной целью данного модуля информатики в условиях реализации ФГОС является формирование у школьника основ алгоритмического мышления.

Исторически сложилось так, что ФГОС основного общего образования претерпевал изменения. Содержательная линия «Алгоритмы и элементы программирования» не всегда была на первом плане в школьном курсе информатики.

Сейчас, изучение данного курса начинается в более раннем возрасте (VIII кл.) и является ведущей содержательной линией в общеобразовательной школе. Огромное количество заданий по данным разделам вынесено на ГИА.

Материал, представленный для изучения в выше перечисленных разделах, достаточно сложен для понимания учащимися и требует многократного практического закрепления.

Изучение учебного модуля «Практикум по основам алгоритмизации и программирования»позволит в полном объеме сформировать УУД учащихся к концу 9 класса.

При изучении учебного модуля**«Практикум по основам алгоритмизации и программирования»**используются следующие **формы текущего контроля успеваемости**: устный (ответы на вопросы) и письменный (самостоятельная работа, проверочная работа, контрольная работа, тест, диктант, практическая работа).

В конце учебного года проводится **промежуточная аттестация** в форме контрольной работы.

Система оценки достижения планируемых результатов

В соответствии с ФГОС ООО система оценки образовательной организации реализует **системно-деятельностный, уровневый и комплексный подходы** к оценке образовательных достижений.

Системно-деятельностный подход к оценке образовательных достижений проявляется в оценке способности учащихся к решению учебно-познавательных и учебно-практических задач. Он обеспечивается содержанием и критериями оценки, в качестве которых выступают планируемые результаты обучения, выраженные в деятельностной форме.

Уровневый подходслужит важнейшей основой для организации индивидуальной работы с учащимися. Он реализуется как по отношению к содержанию оценки, так и к представлению и интерпретации результатов измерений.

Уровневый подход к содержанию оценкиобеспечивается структурой планируемых результатов, в которых выделены три блока: общецелевой, «Выпускник научится» и «Выпускник получит возможность научиться». Достижение планируемых результатов, отнесенных к блоку «Выпускник научится», выносится на итоговую оценку, которая может осуществляться какв ходе обучения, так и в конце обучения, в том числе – в форме государственной итоговой аттестации.

Уровневый подход к представлению и интерпретации результатовреализуется за счет фиксации различных уровней достижения обучающимися планируемых результатов: ***базового уровня и уровней выше и ниже базового.***

Овладение базовым уровнем является достаточным для продолжения обучения и усвоения последующего материала.

Комплексный подход к оценке образовательных достижений реализуется путём

- оценки трёх групп результатов: предметных, личностных, метапредметных (регулятивных, коммуникативных и познавательных универсальных учебных действий);
- использования комплекса оценочных процедур (стартовой, текущей, тематической, промежуточной) как основы для оценки динамики индивидуальных образовательных достижений (индивидуального прогресса) и для итоговой оценки;
- использования контекстной информации (об особенностях обучающихся, условиях и процессе обучения и др.) для интерпретации полученных результатов в целях управления качеством образования;
- использования разнообразных методов и форм оценки, взаимно дополняющих друг друга (стандартизированных устных и письменных работ, проектов, практических работ, самооценки, наблюдения и др.).

Критерии оценивания различных форм работы обучающихся на уроке.

Тематический контроль осуществляется по завершении крупного блока (темы). Он позволяет оценить знания и умения учащихся, полученные в ходе достаточно продолжительного периода работы.

Итоговый контроль осуществляется по завершении каждого года обучения.

Основная форма контроля – тестирование.

Правила при оценивании:

- за каждый правильный ответ начисляется 1 балл;
- за каждый ошибочный ответ начисляется штраф в 1 балл;
- за вопрос, оставленный без ответа (пропущенный вопрос), ничего не начисляется.

Такой подход позволяет добиться вдумчивого отношения к тестированию, позволяет сформировать у школьников навыки самооценки и ответственного отношения к собственному выбору. Тем не менее, учитель может отказаться от начисления штрафных баллов, особенно на начальном этапе тестирования.

Критерии и нормы оценки знаний, умений и навыков учащихся

Оценка устных ответов учащихся

Ответ оценивается отметкой «5» если ученик:

- полно раскрыл содержание материала в объеме, предусмотренном программой и учебником;
- изложил материал грамотным языком в определенной логической последовательности, точно используя терминологию и символику;
- правильно выполнил рисунки, графики, сопутствующие ответу;
- показал умение иллюстрировать теоретические положения конкретными примерами, применять их в новой ситуации при выполнении практического задания;
- продемонстрировал усвоение ранее изученных сопутствующих вопросов, сформированность и устойчивость используемых при ответе умений и навыков;
- отвечал самостоятельно без наводящих вопросов учителя.

Возможны одна-две неточности при освещении второстепенных вопросов или выкладках, которые ученик легко исправил по замечанию учителя; *Ответ оценивается отметкой «4»* если он удовлетворяет в основном требованиям на отметку «5», но при этом имеет один из недостатков:

- в изложении допущены незначительные пробелы, не исказившие содержание ответа;
- допущены один-два недочета при освещении основного содержания ответа, легко исправленные по замечанию учителя;
- допущены ошибка или более двух недочетов при освещении второстепенных вопросов или выкладках, легко исправленные по замечанию учителя.

Отметка «3» ставится в следующих случаях:

- неполно или непоследовательно раскрыто содержание материала, но показано общее понимание вопроса и продемонстрированы умения, достаточные для дальнейшего усвоения программного материала (определенные требованиями к подготовке учащихся).

Отметка «2» ставится в следующих случаях:

- не раскрыто основное содержание учебного материала;
- обнаружено незнание или неполное понимание учеником большей или наиболее важной части учебного материала;
- допущены ошибки в определении понятий, при использовании математической терминологии, в рисунках, чертежах или графиках, в выкладках, которые не исправлены после нескольких

Критерии оценок при выполнении практических заданий:

Оценка «5» - работа выполнена в заданное время, самостоятельно, с соблюдением технологической последовательности, качественно и творчески;

Оценка «4» - работа выполнена в заданное время, самостоятельно, с соблюдением технологической последовательности, при выполнении отдельных операций допущены небольшие отклонения; общий вид аккуратный;

Оценка «3» - работа выполнена в заданное время, самостоятельно, с нарушением технологической последовательности, отдельные операции выполнены с отклонением от образца (если не было на то установки); оформлено небрежно или не закончено в срок;

Оценка «2» - ученик самостоятельно не справился с работой, технологическая последовательность нарушена, при выполнении операций допущены большие отклонения, оформлено небрежно и имеет незавершенный вид.

Критерии оценок для теста:

Оценка «5» - 86% и выше

Оценка «4» - 71% - 85%

Оценка «3» - 50% - 70%

Оценка «2» - 49% и ниже

Критерии оценок для творческого проекта:

- эстетичностьоформления,
- содержание, соответствующеетемеработы,
- полная и достоверная информация по теме,
- отражение всех знаний и умений учащихся в данной программе,
- актуальность выбранной темы в учебно-воспитательном процессе.

В настоящей программе учтено, что сегодня, в соответствии с Федеральным государственным стандартом начального образования, учащиеся к концу начальной школы должны обладать ИКТ-компетентностью, достаточной для дальнейшего обучения. Курс информатики в 7-9 классах, опирается на опыт постоянного применения ИКТ, уже имеющийся у учащихся, дает теоретическое осмысление, интерпретацию и обобщение этого опыта.

При организации занятий школьников по информатике и информационным технологиям необходимо использовать различные методы и средства обучения с тем, чтобы с одной стороны, свести работу за ПК к регламентированной норме; с другой стороны, достичь наибольшего педагогического эффекта.

На уроках параллельно применяются общие и специфические **методы**, связанные с применением средств ИКТ:

- словесные методы обучения (рассказ, объяснение, беседа, работа с учебником, рабочей тетрадью);

- наглядные методы (наблюдение, иллюстрация, демонстрация наглядных пособий, презентаций);
- практические методы (устные и письменные упражнения, практические работы за ПК);
- проблемноеобучение;
- методпроектов;
- ролевойметод.

Форма организации образовательного процесса: классно-урочная система, дистанционная.

Применение на уроках интерактивных форм работы собучающимися является ведущей формой организации учебной деятельности учащихся. На уроках в соответствии с Программой формирования/развития УУД и РПВ используются следующие формы совместной деятельности учащихся: интеллектуальных игр, стимулирующих познавательную мотивацию учащихся; дидактического театра, где полученные на уроке знания обыгрываются в театральных постановках; дискуссий, которые дают учащимся возможность приобрести опыт ведения конструктивного диалога; групповой работы или работы в парах, которые учат учащихся командной работе и взаимодействию с другими учащимися.

Включение в урок игровых процедур, которые помогают поддержать мотивацию детей к получению знаний, налаживанию позитивных межличностных отношений в классе, помогают установлению доброжелательной атмосферы во время урока: «Зигзаг», «Учебное задание-проект», «Найди ошибку», «Продолжи» и др.

Применение на уроках организации приемов шевства – заданий на помощь и взаимовыручку.

Формыуроков:

- уроки «открытия» новогознания;
- урокирефлексии;
- урокиобщеметодологическойнаправленности;
- урокиразвивающегоконтроля.

Технологии, используемые в обучении:

- развивающегообучения,
- обучения в сотрудничестве,
- проблемногообучения,
- развитияисследовательскихнавыков,
- информационно-коммуникационные,

- здоровьесбережения и т. д.

На изучение учебного модуля «Практикум по основам алгоритмизации и программирования»на уровне основного общего образования отводится 1 учебный час в неделю в 8-х классах, всего 36 часов.

В соответствии с требованиями Федерального государственного образовательного стандарта основного общего образования учебный модуль «Практикум по основам алгоритмизации и программирования» является частью предмета«Информатика и входит в предметную область «Естественно - научные предметы».

2.Планируемые результаты освоения учебного модуля

Рабочая программа обеспечивает формирование личностных, метапредметных и предметных результатов.

Личностные результаты - это сформировавшаяся в образовательном процессе система ценностных отношений учащихся к себе, другим участникам образовательного процесса, самому образовательному процессу, объектам познания, результатам образовательной деятельности. Основными личностными результатами, формируемыми при изучении информатики в основной школе, являются:

- наличие представлений об информации как важнейшем стратегическом ресурсе развития личности, государства, общества;
- понимание роли информационных процессов в современном мире;
- владение первичными навыками анализа и критической оценки получаемой информации;
- ответственное отношение к информации с учетом правовых и этических аспектов ее распространения;
- развитие чувства личной ответственности за качество окружающей информационной среды;
- способность увязать учебное содержание с собственным жизненным опытом, понять значимость подготовки в области информатики и ИКТ в условиях развития информационного общества;
- готовность к повышению своего образовательного уровня и продолжению обучения с использованием средств и методов информатики и ИКТ;
- способность и готовность к общению и сотрудничеству со сверстниками и взрослыми в процессе образовательной, общественно-полезной, учебно-исследовательской, творческой деятельности;

- способность и готовность к принятию ценностей здорового образа жизни за счет знания основных гигиенических, эргономических и технических условий безопасной эксплуатации средств ИКТ.

Метапредметные результаты - освоенные обучающимися на базе одного, нескольких или всех учебных предметов способы деятельности, применимые как в рамках образовательного процесса, так и в других жизненных ситуациях. Основные метапредметные результаты, формируемые при изучении информатики в основной школе, включают в себя владение:

- общепредметными понятиями «объект», «система», «модель», «алгоритм», «исполнитель» и др.;
- информационно-логическими умениями: определять понятия, создавать обобщения, устанавливать аналогии, классифицировать, самостоятельно выбирать основания и критерии для классификации, устанавливать причинно-следственные связи, строить логическое рассуждение, умозаключение (индуктивное, дедуктивное и по аналогии) и делать выводы;
- умениями самостоятельно планировать пути достижения целей; соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; оценивать правильность выполнения учебной задач;
- основами самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности;
- основными универсальными умениями информационного характера: постановка и формулирование проблемы; поиск и выделение необходимой информации, применение методов информационного поиска; структурирование и визуализация информации; выбор наиболее эффективных способов решения задач в зависимости от конкретных условий; самостоятельное создание алгоритмов деятельности при решении проблем творческого и поискового характера;
- ИКТ-компетентностью - широким спектром умений и навыков использования средств информационных и коммуникационных технологий для сбора, хранения, преобразования и передачи различных видов информации, навыками создания личного информационного пространства (обращение с устройствами ИКТ; фиксация изображений и звуков; создание письменных сообщений, графических объектов, музыкальных и звуковых сообщений; создание, восприятие и использование

гипермедиасообщений; коммуникация и социальное взаимодействие; поиск и организация хранения информации; анализ информации).

Основные предметные результаты, формируемые при изучении учебного модуля в основной школе, включают в себя:

- формирование информационной и алгоритмической культуры; формирование представления о компьютере как универсальном устройстве обработки информации; развитие основных навыков и умений использования компьютерных устройств;
- формирование представления об основных изучаемых понятиях: информация, алгоритм, модель - и их свойствах;
- развитие алгоритмического мышления, необходимого для профессиональной деятельности в современном обществе; развитие умений составить и записать алгоритм для конкретного исполнителя; формирование знаний об алгоритмических конструкциях, логических значениях и операциях; знакомство с одним из языков программирования и основными алгоритмическими структурами - линейной, условной и циклической;

Предметные результаты

Математические основы информатики

8 класс

Ученик научится:

- записывать в двоичной системе целые числа от 0 до 256;
- составлять логические выражения с операциями И, ИЛИ, НЕ; определять значение логического выражения; строить таблицы истинности;

Ученикполучитвозможность:

- переводить небольшие десятичные числа из восьмеричной и шестнадцатеричной системы счисления в десятичную систему счисления;
- научиться решать логические задачи с использованием таблиц истинности;
- научиться решать логические задачи путем составления логических выражений и их преобразования с использованием основных свойств логических операций;

Алгоритмы и элементы программирования.

8 класс

Ученик научится:

- понимать смысл понятия «алгоритм» и широту сферы его применения; анализировать предлагаемые последовательности команд на предмет наличия у них таких свойств

алгоритма, как дискретность, детерминированность, понятность, результативность, массовость;

- оперировать алгоритмическими конструкциями «следование», «ветвление», «цикл» (подбирать алгоритмическую конструкцию, соответствующую той или иной ситуации; переходить от записи алгоритмической конструкции на алгоритмическом языке к блок-схеме и обратно);
- понимать термины «исполнитель», «формальный исполнитель», «среда исполнителя», «система команд исполнителя» и др.; понимать ограничения, накладываемые средой исполнителя и системой команд, на круг задач, решаемых исполнителем;
- исполнять линейный алгоритм для формального исполнителя с заданной системой команд;
- составлять линейные алгоритмы, число команд в которых не превышает заданное;
- ученик научится исполнять записанный на естественном языке алгоритм, обрабатывающий цепочки символов;
- исполнять линейные алгоритмы, записанные на алгоритмическом языке;
- исполнять алгоритмы с ветвлениями, записанные на алгоритмическом языке;
- понимать правила записи и выполнения алгоритмов, содержащих цикл с параметром или цикл с условием продолжения работы;
- определять значения переменных после исполнения простейших циклических алгоритмов, записанных на алгоритмическом языке;
- разрабатывать и записывать на языке программирования короткие алгоритмы, содержащие базовые алгоритмические конструкции.

Ученикполучитвозможность:

- исполнять алгоритмы, содержащие ветвления и повторения, для формального исполнителя с заданной системой команд;
- составлять все возможные алгоритмы фиксированной длины для формального исполнителя с заданной системой команд;
- определять количество линейных алгоритмов, обеспечивающих решение поставленной задачи, которые могут быть составлены для формального исполнителя с заданной системой команд;
- подсчитывать количество тех или иных символов в цепочке символов, являющейся результатом работы алгоритма;
- по данному алгоритму определять, для решения какой задачи он предназначен;
- исполнять записанные на алгоритмическом языке циклические алгоритмы обработки одномерного массива чисел (суммирование всех элементов массива; суммирование

элементов массива с определенными индексами; суммирование элементов массива с заданными свойствами; определение количества элементов массива с заданными свойствами; поиск наибольшего/ наименьшего элементов массива и др.);

- разрабатывать в среде формального исполнителя короткие алгоритмы, содержащие базовые алгоритмические конструкции;
- разрабатывать и записывать на языке программирования эффективные алгоритмы, содержащие базовые алгоритмические конструкции.

3. Содержание учебного модуля

Математическиеосновыинформатики

Системысчисления

Позиционные и непозиционные системы счисления. Примеры представления чисел в позиционных системах счисления.

Основание системы счисления. Алфавит (множество цифр) системы счисления. Количество цифр, используемых в системе счисления с заданным основанием. Краткая и развернутая формы записи чисел в позиционных системах счисления.

Двоичнаясистемасчисления,записьцелыхчиселвпределахот0до 1024. Перевод натуральных чисел из десятичной системы счисления в двоичнуюиз двоичнойв десятичную.

Восьмеричная и шестнадцатеричная системы счисления. Перевод натуральных чисел из десятичной системы счисления в восьмеричную, шестнадцатеричную и обратно.

Перевод натуральных чисел из двоичной системы счисления в восьмеричную и шестнадцатеричную и обратно.

Арифметические действия в системах счисления.

Элементыкомбинаторики,теориимножествиматематической логики

Высказывания. Простые и сложные высказывания. Диаграммы Эйлера-Венна. Логические значения высказываний. Логические выражения. Логические операции: «и» (конъюнкция, логическое умножение), «или» (дизъюнкция, логическое сложение), «не» (логическое отрицание). Правила записи логических выражений. Приоритеты логических операций.

Таблицыистинности. Построениетаблицистинностидлялогических выражений.

Логические операции следования (импликация) и равносильности (эквивалентность).Свойства логических операций. Законы алгебры логики. Использование таблиц

истинности для доказательства законов алгебры логики. Логические элементы. Схемы логическихэлементовиихфизическая(электронная)реализация.Знакомство слогическимиосновамикомпьютера.

Алгоритмы и элементы программирования

Исполнители и алгоритмы. Управление исполнителями

Исполнители. Состояния, возможные обстановки и система команд исполнителя; команды-приказы и команды-запросы; отказ исполнителя. Необходимость формального описания исполнителя. Ручное управление исполнителем.

Алгоритм как план управления исполнителем (исполнителями). Алгоритмический язык (язык программирования) – формальный язык для записи алгоритмов. Программа – запись алгоритма на конкретном алгоритмическом языке. Компьютер – автоматическое устройство, способное управлять по заранее составленной программе исполнителями, выполняющими команды. Программное управление исполнителем. *Программное управление самодвижущимся роботом.*

Словесное описание алгоритмов. Описание алгоритма с помощью блок-схем. Отличие словесного описания алгоритма, от описания на формальном алгоритмическом языке.

Системы программирования. Средства создания и выполнения программ.

Понятие об этапах разработки программ и приемах отладки программ.

Управление. Сигнал. Обратная связь. Примеры: компьютер и управляемый им исполнитель (в том числе робот); компьютер, получающий сигналы от цифровых датчиков в ходе наблюдений и экспериментов, и управляющий реальными (в том числе движущимися) устройствами.

Алгоритмические конструкции

Конструкция «следование». Линейный алгоритм. Ограниченность линейных алгоритмов: невозможность предусмотреть зависимость последовательности выполняемых действий от исходных данных.

Конструкция «ветвление». Условный оператор: полная и неполная формы.

Выполнение и невыполнение условия (истинность и ложность высказывания). Простые и составные условия. Запись составных условий.

Конструкция «повторения»: циклы с заданным числом повторений, с условием выполнения, с переменной цикла. *Проверка условия выполнения цикла до начала выполнения тела цикла и после выполнения тела цикла: постусловие и предусловие цикла. Инвариант цикла.*

Запись алгоритмических конструкций в выбранном языке программирования.

Примеры записи команд ветвления и повторения и других конструкций в различных алгоритмических языках.

Разработка алгоритмов и программ

Оператор присваивания. *Представление о структурах данных.*

Константы и переменные. Переменная: имя и значение. Типы переменных: целые, вещественные, *символьные, строковые, логические*. Табличные величины (массивы). Одномерные массивы. *Двумерные массивы*.

Примеры задач обработки данных:

- нахождение минимального и максимального числа из двух, трех, четырех данных чисел;
- нахождение всех корней заданного квадратного уравнения;
- заполнение числового массива в соответствии с формулой или путем ввода чисел;
- нахождение суммы элементов данной конечной числовой последовательности или массива;
- нахождение минимального (максимального) элемента массива.

Знакомство с алгоритмами решения этих задач. Реализации этих алгоритмов в выбранной среде программирования.

Составление алгоритмов и программ по управлению исполнителями Робот, Черепашка, Чертежник и др.

Знакомство с постановками более сложных задач обработки данных и алгоритмами их решения: сортировка массива, выполнение поэлементных операций с массивами; обработка целых чисел, представленных записями в десятичной и двоичной системах счисления, нахождение наибольшего общего делителя (алгоритм Евклида).

Понятие об этапах разработки программ: составление требований к программе, выбор алгоритма и его реализация в виде программы на выбранном алгоритмическом языке, отладка программы с помощью выбранной системы программирования, тестирование.

Простейшие приемы диалоговой отладки программ (выбор точки останова, пошаговое выполнение, просмотр значений величин, отладочный вывод).

Знакомство с документированием программ. *Составление описание программы по образцу.*

Анализ алгоритмов

Сложность вычисления: количество выполненных операций, размер используемой памяти; их зависимость от размера исходных данных. Примеры коротких программ, вы-

полняющих много шагов по обработке небольшого объема данных; примеры коротких программ, выполняющих обработку большого объема данных.

Определение возможных результатов работы алгоритма при данном множестве входных данных; определение возможных входных данных, приводящих к данному результату. Примеры описания объектов и процессов с помощью набора числовых характеристик, а также зависимостей между этими характеристиками, выражаемыми с помощью формул.

4.Тематическое планирование учебного модуля

«Практикум по основам алгоритмизации и программирования»

Предметноесодержаниетемыуроков	Кол – во часов на изучение каждой темы	Этнокультурная составляющая (количество часов с указанием темы ЭКС)	Практическаячасть	Основныевидыучебнойдеятельности	Организация обсуждения учащимися ценностных аспектов изучаемых явлений, организация работы с социально-значимой информацией	Текстыдлячтения	Кейсы для организации проектной и исследовательской деятельности
8 класс **(36 часов – 1 час в неделю).** **1 раздел «Теоретические основы информатики» - 8 часов**							
Тема № 1 «Решение задач с римскими цифрами»	1 час			***Аналитическая деятельность:*** - выявляет различие в унарных, позиционных и непозиционных системах счисления; - выявляет общее и отличия в разных позиционных системах счисления; - анализирует логическую структуру высказываний. ***Практическая деятельность:*** - переводит небольшие (от 0 до			
Тема № 2 «Перевод двоичных чисел в десятичную систему счисления и обратно»	1 час		Решение задач по теме «Двоичная система счисления»				
Тема № 3«Перевод восьмеричных и шестнадцатеричных чисел в десятичную систему счисления и обратно»	1 час		Практическая работа «Системы счисления»		Как числа, записанные в различных системах счисления, хранятся и обрабатываются в памяти компьютера?		Кейс № 1 «Системы счисления»

Тема № 4 «Арифметические действия в q-ичных системе счисления»	1 час		Решение задач по теме «Арифметика в двоичной системе счисления»	1024) целые числа из десятичной системы счисления в двоичную (восьмеричную,шестнадцатеричную) и обратно; - выполняет операции сложения и умножения над небольшими двоичными числами; - записывает вещественные числа в естественной и нормальной форме; - строит таблицы истинности для логических выражений; - вычисляет истинностное значение логического выражения. -кодирует и декодирует сообщения по известным правилам кодирования; - определяет количество различных символов, которые			
Тема № 5 «Нахождение значений логических выражений»	1 час		Решение задач по теме «Базовые логические операции»		С какой целью можно использовать логические операции применительно к различным видам информации?		
Тема № 6 «Применение алгебры логики для решения текстовых логических задач»	1 час		Решениелогических задач				
Тема № 7 «Заполнение таб-	1 час		Решение задач по теме				

лицы логическими значениями»			«Построение таблиц истинности для логических выражений	могут быть закодированы с помощью двоичного кода фиксированной длины (разрядности);			
Тема № 8 «Составление логических схем»	1 час		Компьютерный тест по теме «Логические элементы»	- определяет разрядность двоичного кода, необходимого для кодирования всех символов алфавита заданной мощности;	Почему необходимо уметь строить логические схемы?		
2 раздел «Алгоритмы и программирование» - 28 часов							
Тема № 9«Знакомство со средой алгоритмического языка Кумир. Исполнитель Черепашка»	1 час		Составление простейших программ	***Аналитическая деятельность:*** - определяет по блок-схеме, для решения какой задачи предназначен данный алгоритм; - анализирует изменение значений величин при пошаговом выполнении алгоритма; - определяет по выбранному методу решения за-			
Тема № 10 «Исполнитель Робот. Составление простейших программ»	1 час		Составление простейших программ				
Тема № 11«Алгоритмы решения задач на переливание. Исполнитель Водолей»	1 час		Решение задач на переливание.			История развития понятия алгоритма	

Тема № 12«Составление алгоритмов для учебного исполнителя Кузнечик»	1 час		Составление простейших программ	дачи, какие алгоритмические конструкции могут войти в алгоритм; - сравнивает различные алгоритмы решения одной задачи. ***Практическая деятельность:*** - исполняет готовые алгоритмы для конкретных исходных данных; - преобразовывает запись алгоритма с одной формы в другую; - строит цепочки команд, дающих нужный результат при конкретных исходных данных для исполнителя арифметических действий; - строит цепочки команд, дающих нужный результат при конкретных исходных данных для исполнителя, преобразующего			
Тема № 13«Исполнитель Чертёжник. Общие сведения. Описание команд»	1 час		Описание команд		Какие командыдолжны быть у исполнителя Чертежник?		
Тема № 14 «Выполнение простейших чертежей»	1 час		Выполнение простейших чертежей				
Тема № 15 «Решение задач с циклом системе КуМир»	1 час		Написание программ				
Тема № 16 «Создание сложного орнамента»	1 час		Создание сложного орнамента				
Тема № 17 «Ветвление в языке КуМир»	1 час		Написание программ				
Тема № 18 «Решение задач с вложенными ветвлениями и циклами»	1 час		Решение задач				
Тема № 19 «Решение экзаменационных задач с	1 час		Решение задач				Кейс № 2 «Алгоритмы»

использованием программной среды КуМир»				строки символов; - строит арифметические, строковые, логические выражения и вычислять их значения.			
Тема № 20 «Организация ввода и вывода данных на Паскаль»	1 час		Разработка простой программы	***Аналитическая деятельность:*** - анализирует готовые программы; - определяет по программе, для решения какой задачи она предназначена; - выделяет этапы решения задачи на компьютере. ***Практическая деятельность:*** - программирует линейные алгоритмы, предполагающие вычисление арифметических, строковых и логических выражений; - разрабатывает программы, содержащие оператор/операторы		Компьютерное программирование	Кейс № 3 «Основные понятия языка Паскаль»
Тема № 21 «Программирование линейных алгоритмов»	1 час		Разработка программ с линейным алгоритмом		В чем сходство и отличие структуры алгоритма со структурой программы на языке Паскаль?		
Тема № 22 «Арифметика Паскаль»	1 час		Запись арифметических выражений				
Тема № 23 «Строковый тип данных»	1 час		Разработка простой программы				
Тема № 24 «Условный оператор языка Паскаль»	1 час		Разработка программ с применением алгоритмов ветвления				
Тема № 25 «Программирование циклов с заданным усло-	1 час		Разработка циклических алгоритмов				

вием продолжения работы»				ветвления (решение линейного неравенства, решение квадратного уравнения и пр.), в том числе с использованием логических операций; - разрабатывает программы, содержащие оператор (операторы) цикла			
Тема № 26 «Программирование циклов с заданным условием окончания работы»	1 час		Разработка циклических алгоритмов				
Тема № 27 «Программирование циклов с заданным числом повторений»	1 час		Разработка циклических алгоритмов				
Тема № 28 «Вызов графического модуля. Основные графические операторы»	1 час		Запуск программ с использованием графического модуля. Построение геометрических фигур				
Тема № 29«Стандартные цвета»	1 час		Построение шахматной доски				
Тема № 30«Линейная функция»	1 час		«Построение графика линейной функции»				
Тема № 31«Степенные функции».	1 час		Аргументы оператора power и trunk				

Тема № 32«Тригонометрические функции».	1 час		Аргументы оператора sin и cos				
Тема № 33«Организация совместного вывода графики и текста на экран»	1 час		Организация совместного вывода графики и текста на экран		Чем отличается процедура OutText от процедуры Write?		
Тема № 34 «Промежуточная аттестация»	1 час		Итоговая контрольная работа за курс 8 класса				

Приложение 1

Тексты для чтения

История развития понятия алгоритма

Понятие алгоритма является одним из основных понятий современной математики. Еще на самых ранних ступенях развития математики (Древний Египет, Вавилон, Греция) в ней стали возникать различные вычислительные процессы чисто механического характера. С их помощью искомые величины ряда задач вычислялись последовательно из исходных величин по определенным правилам и инструкциям. Со временем все такие процессы в математике получили название алгоритмов.

Термин алгоритм происходит от имени средневекового узбекского математика Аль-Хорезми, который еще в IX в. дал правила выполнения четырех арифметических действий в десятичной системе счисления.

Вплоть до 30-х годов прошлого века понятие алгоритма имело скорее методологическое, чем математическое значение. Под алгоритмом понимали конечную совокупность точно сформулированных правил, которые позволяют решать те или иные классы задач. Однако в этом определении не содержится точной характеристики того, что следут понимать под классом задач и под правилами их решения. В течение длительного времени математики довольствовались этим определением, поскольку общей теории алгоритмов фактически не существовало. Однако, практически не было серьезных случаев, когда математики разошлись бы во мнении относительно того, является ли алгоритмом тот или иной конкретно заданный процесс.

Положение существенно изменилось, когда на первый план выдвинулись такие алгоритмические проблемы, положительное решение которых было сомнительным. Действительно, одно дело доказать существование алгоритма, другое – его отсутствие. Первое можно сделать путем фактического описания процесса, решающего задачу. В этом случае достаточно и интуитивного понятия алгоритма, чтобы удостовериться в том, что описанный процесс есть алгоритм. Доказать несуществование алгоритма таким путем невозможно. Для этого надо точно знать, что такое алгоритм.

В двадцатых годах прошлого века задача такого определения понятия алгоритма стала одной из центральных математических проблем. Решение ее было получено в середине тридцатых годов в работах известных математиков: Гильберта, Геделя, Черча, Клини, Поста и Тьюринга.

В 50-е годы прошлого столетия существенный вклад в развитие теории алгоритмов внесли работы Колмогорова и Маркова. Формальные модели алгоритмов Поста, Тьюринга

и Черча, равно как и модели Колмогорова и Маркова, оказались эквивалентными в том смысле, что любой класс проблем, разрешимых в одной модели, разрешим и в другой.

Хотя первоначально теория алгоритмов возникла в связи с внутренними потребностями теоретической математики (математическая логика, алгебра, геометрия и т.д. остаются и сегодня одной из основных областей приложения теории алгоритмов), другая область ее применения возникла в 40-х годах в связи с созданием быстродействующих электронных вычислительных и управляющих машин. Появление ЭВМ способствовало развитию теории алгоритмов, вызвало к жизни разделы этой теории , имеющие ярко выраженную прикладную направленность. Это, прежде всего, алгоритмические системы и алгоритмические языки, являющиеся основой современной теории программирования для универсальных ЭВМ, и способы точного описания отображений, реализуемых цифровыми автоматами.

Теория алгоритмов оказалась тесно связанной и с рядом областей лингвистики, экономики, физиологии мозга и психологии, философии, естествознания. Примером одной из задач этой области может служить точное описание алгоритмов, реализуемых человеком в процессе умственной деятельности.

Вопросы:

1.Какие математические процессы получили название алгоритмов?

2. Когда было сформулировано точное определение понятия алгоритма?

3.Что повлияло на развитие алгоритмических языков?

4.Какие сферы активно используют теорию алгоритмов?

Компьютерное программирование

Программирование - это процесс подготовки набора закодированных инструкций, которые позволяют компьютеру решать определенные проблемы или выполнять определенные функции. Суть компьютерного программирования - это кодирование программы для компьютера с помощью алгоритмов. Дело в том, что любая проблема выражается математически, содержит формулы, уравнения и расчеты ». Но компьютер не может управлять формулами, уравнениями и вычислениями. Любая проблема должна быть специально обработана, чтобы компьютер ее понял, то есть - закодирован или запрограммирован. Фаза, на которой пишутся компьютерные программы системы, называется фазой разработки.

Программы представляют собой списки инструкций, которым будет следовать блок управления центрального процессора (ЦП). Инструкции программы должны быть полными и в соответствующей последовательности, иначе будут получены неправильные

ответы. Для защиты от этих логических ошибок и документирования логического подхода программы необходимо разработать логические планы.

Есть два распространенных метода планирования логики программы. Первый метод - это блок-схема. Блок-схема - это план в форме графического или графического представления, в котором используются заранее определенные символы для иллюстрации логики программы. Таким образом, это «картина» логических шагов, которые должен выполнить компьютер. Каждая из предопределенных форм символа обозначает общую операцию. Форма символа передает характер общей операции, а детали записываются внутри символа. Пластиковая или металлическая направляющая, называемая шаблоном, используется для облегчения рисования символов. Второй метод планирования логики программы называется псевдокодом. Псевдокод - это имитация реальных программных инструкций. Это позволяет создавать программную структуру без бремени правил программирования. Псевдокод занимает меньше времени для профессионального программиста, чем создание блок-схем. Он также подчеркивает нисходящий подход к структуре программы. Псевдокод имеет три основные структуры: последовательность, решение и логику цикла. С помощью этих трех структур может быть выражена любая необходимая логика.

Вопросы:

1. В чем суть программирования?
2. Что делать с проблемой перед обработкой на компьютере?
3. Каковы основные методы планирования логики программы?
4. Что такое шаблон и для чего он используется?
5. Каковы основные структуры псевдокода?

Приложение 2

Кейсы для организации проектной и исследовательской деятельности

Кейс № 1 «Системы счисления»

Ежегодно в ГБПОУ «Сергиевский губернский техникум» проходит декада математических и общих естественнонаучных дисциплин. В рамках указанной декады преподаватели проводят олимпиаду по математическим дисциплинам. В 2013-2014 учебном году среди олимпиадных заданий по дисциплине «Математика и информатик» были следующие:

1) Восстановите цифры двоичной системы счисления, на месте которых в арифметических выражениях стоит знак "*".

а) $**0*0*1**1_2 + 10111*10**_2 = 100*1*00010_2$

б) $***0**00_2 - 11*11*11_2 = 1101*1_2$

2) В классе 1000112 учеников. 1111002% из них учатся на хорошо и отлично.

Сколько учеников учатся на хорошо и отлично?

Студенты первокурсники плохо справились с данными заданиями, несмотря на то, что задания относились к уровню низкой сложности.

Что помешало студентам решить задания?

Вопросы и задания:

1.Составьте конспект по теме «Позиционные системы счисления и действия над числами в позиционных системах счисления, отличных от десятичной».

2. Выполните задания и изложите решение, включая основания для него (письменная форма).

Кейс № 2«Алгоритмы»

Лекция 1. Этапы решения задач на компьютере. Алгоритм и его свойства. Способы записи алгоритмов.

На первом этапе участвует человек, хорошо представляющий предметную область задачи.

Он должен четко определить цель задачи, дать словесное описание содержания задачи и предложить общий подход к её решению.

Для задачи вычисления суммы двух целых чисел человек, знающий, как складываются числа, может описать задачу следующим образом: ввести два целых числа, сложить их и вывести сумму в качестве результата решения задачи.

Цель второго этапа – создать такую математическую модель решаемой задачи, которая может быть реализована в компьютере.

Существует целый ряд задач, где математическая постановка сводится к простому перечислению формул и логических условий.

Этот этап тесно связан с первым этапом, и его можно отдельно рассматривать, однако возможно, что для полученной модели известны несколько методов решения, и тогда предстоит выбрать лучший.

Для вышеописанной задачи данный этап сведется к следующему: введенные в компьютер числа запомним в памяти под именами A и B, затем вычислим значение суммы этих чисел по формуле A+B, и результат запомним в памяти под именем S.

На третьем этапе на основе математического описания необходимо разработать алгоритм решения.

Составление программы на четвертом этапе обеспечивает возможность выполнения алгоритма и соответственно поставленной задачи исполнителем – компьютером.

Во многих задачах при программировании на алгоритмическом языке часто пользуются заменой блока алгоритма на один лил несколько операторов, введением новых блоков, заменой одних блоков другими. Программа и исходные данные вводятся в ЭВМ с клавиатуры с помощью редактора текстов, и для постоянного хранения осуществляется их запись на гибкий или жесткий магнитный диск.

На шестом этапе происходят выполнение алгоритма с помощью ЭВМ, поиск и исключение ошибок.

При этом программисту приходится выполнять рутинную работу по проверке работы программы, поиску и исключению ошибок, и поэтому для сложных программ этот часто требует гораздо больше временит и сил, чем написание первоначального текста программы.

Далее программист запускает программу и задает исходные данные, требуемые по условию задачи.

Алгоритм — это понятное и точное указание исполнителю совершить последовательность действий, направленных на решение поставленной задачи.

Термин имеет интересное историческое происхождение. В IX веке великий узбекский математикаль-Хорезми разработал правила арифметических действий над десятичными числами. Совокупность этих правил в Европе стали называть "алгоритм". Впоследствии слово трансформировалось до известного нам сейчас вида и, кроме того, расширило свое значение: алгоритмом стали называть любую последовательность действий (не только арифметических), которая приводит к решению той или иной задачи. Можно сказать, что понятие вышло за рамки математики и стало применяться в самых различных областях.

Для того чтобы произвольное описание последовательности действий было алгоритмом, оно должно обладать следующими свойствами.

Дискретность

Процесс решения задачи должен быть разбит на последовательность отдельных шагов, каждый из которых называется командой. Примером команд могут служить пункты инструкции, нажатие на одну из кнопок пульта управления, рисование графического примитива (линии, дуги и т.п.), оператор языка программирования. Наиболее существенным здесь является тот факт, что алгоритм есть последовательность четко выделенных пунктов, — такие "прерывные" объекты в науке принято называть дискретными.

Понятность

Каждая команда алгоритма должна быть понятна тому, кто исполняет алгоритм; в противном случае эта команда и, следовательно, весь алгоритм в целом не могут быть вы-

полнены. Данное требование можно сформулировать более просто и конкретно. Составим полный список команд, которые умеет делать исполнитель алгоритма, и назовем его системой команд исполнителя (СКИ). Тогда понятными будут являться только те команды, которые попадают в этот список. Именно из такой формулировки становится ясно, почему компьютер такой "привередливый" при приеме введенных в него команд: даже если неверно написана всего одна буква, команда уже не может быть обнаружена в СКИ.

Команды, образующие алгоритм (или, можно сказать, входящие в СКИ), должны быть предельно четкими и однозначными. Их результат не может зависеть от какой-либо дополнительной информации извне алгоритма. Сколько бы раз вы не запускали программу, для одних и тех же исходных данных всегда будет получаться один и тот же результат.

Результативность

Результат выполнения алгоритма должен быть обязательно получен, т.е. правильный алгоритм не может обрываться безрезультатно из-за какого-либо непреодолимого препятствия в ходе выполнения. Кроме того, любой алгоритм должен завершиться за конечное число шагов.

Корректность

Любой алгоритм создан для решения той или иной задачи, поэтому нам необходима уверенность, что это решение будет правильным для любых допустимых исходных данных. Указанное свойство алгоритма принято называть его корректностью

Массовость

Алгоритм имеет смысл разрабатывать только в том случае, когда он будет применяться многократно для различных наборов исходных данных.

Способы записи алгоритмов:

1. Словесный способ записи
2. Запись на алгоритмическом языке
3. Запись на языке блок-схем
4. Запись на языке программирования

Задания:

1. Которые из документов являются алгоритмами?
   1) Программа телепередач
   2) Правило правописания приставок, оканчивающихся на з, с
   3) Кулинарный рецепт приготовления блюда
   4) Инструкция по сборке проданного в разобранном виде шкафа
   5) Порядок набора международного телефонного номера

6) Настенный календарь на текущий год
7) Каталог книг в библиотеке
8) Рецепт приготовления клея
9) Правила игры в футбол
10) Политическая карта мира
11) Телефонный справочник
12) Файл Readme, содержащий информацию об инсталляции программы
13) Вычисление корней квадратного уравнения
14) Решение шахматной задачи
15) Инструкция по сборке видеомагнитофона
16) Морфологический разбор слова

2. Установите соответствие между свойствами алгоритма и ситуациями, в которых эти свойства были нарушены. Переставьте строки второго столбца, чтоб они соответствовали строкам первого. Запишите в ответе последовательность из пяти букв, соответствующую последовательности названий строк второго столбца.

1. Результативность	А. Компьютер посчитал результат вычислений, но не вывел его на экран.
2.Конечность	Б. Программист составил программу для одного конкретного значения исходных данных.
3. Массовость	В. В алгоритме в одной из строк программист написал «И так далее».
4. Дискретность	Г. В программе для Черепашки кто-то вместо команды НАПРАВО написал ВПРАВО.
5. Понятность	Д. В инструкции по приготовлению горячего блюда содержался пункт «Ждать, пока закипит», но оказалась ошибочно пропущена строка «Включить плиту».

3. Вы обучаете Светлого Робота перемещаться по клетчатому полю, в котором между соседними клетками могут располагаться стены.

Начальное положение Светлого Робота – клетка a1.

Робот умеет выполнять команды:

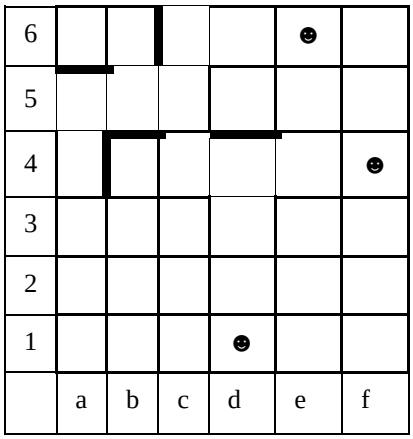
(1) – передвигается на одну клетку вверх;

(2) - передвигается на одну клетку вниз;

(3) - передвигается на одну клетку вправо;

(4) - передвигается на одну клетку влево.

В какую клетку попадёт Робот, если выполнит команды: 1111314?



Лекция 2. Виды алгоритмов. Запись алгоритмов с помощью блок-схем

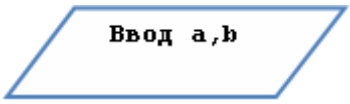
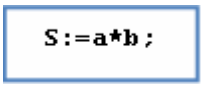
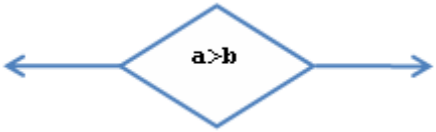
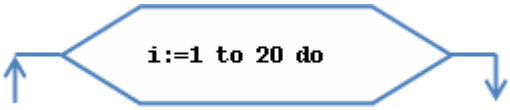
Виды алгоритмов

Название	Определение	Примеры
Линейный (последовательный)	Действия выполняются однократно в заданном порядке	Приготовление одного бутерброда
Циклический	Действия повторяются указанное число раз в заданном порядке	Создание рисунков из повторяющихся элементов
Разветвляющийся (условный)	В зависимости от условия выполняется либо одна, либо другая последовательность действий	Правило правописание приставок, оканчивающихся на з, с

Для более наглядного представления алгоритма используется графический способ. Существует несколько способов графического описания алгоритмов. Наиболее широко используемым на практике графическим описанием алгоритмов является использование блок-схем. Несомненное достоинство блок схем – наглядность и простота записи алгоритма.

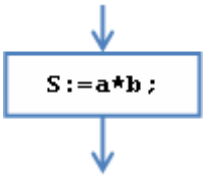
Каждому действию алгоритма соответствует геометрическая фигура (блочный символ). Перечень наиболее часто употребляемых символов приведен в таблице:

Название символа	Обозначение и пример заполнения	Пояснения
Пуск-останов	Начало	Начало, завершение алгоритма или подпрограммы

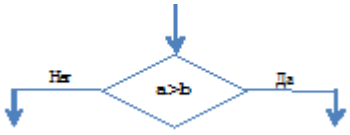
Название символа	Обозначение и пример заполнения	Пояснения
Ввод-вывод данных		Ввод исходных данных или вывод результатов
Процесс		Внутри прямоугольника записывается действие, например, расчетная формула
Решение		Проверка условия, в зависимости от которого меняется направление выполнения алгоритма
Модификация		Организация цикла
Предопределенный процесс		Использование ранее созданных подпрограмм
Комментарий		Пояснения

Пояснения:

- блок Процесс обозначает вычислительный процесс и применяется для обозначения действия или последовательности действий, изменяющих значения переменных или данных

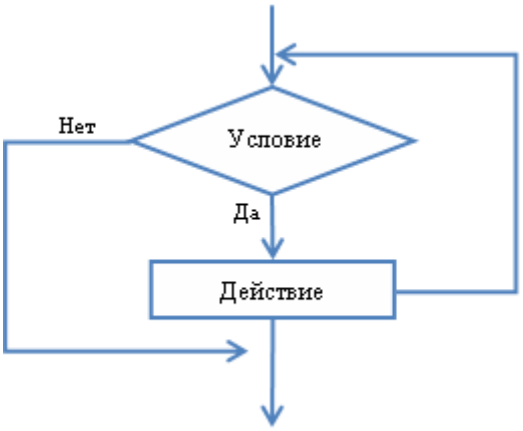


- блок Решение обозначает проверку условия



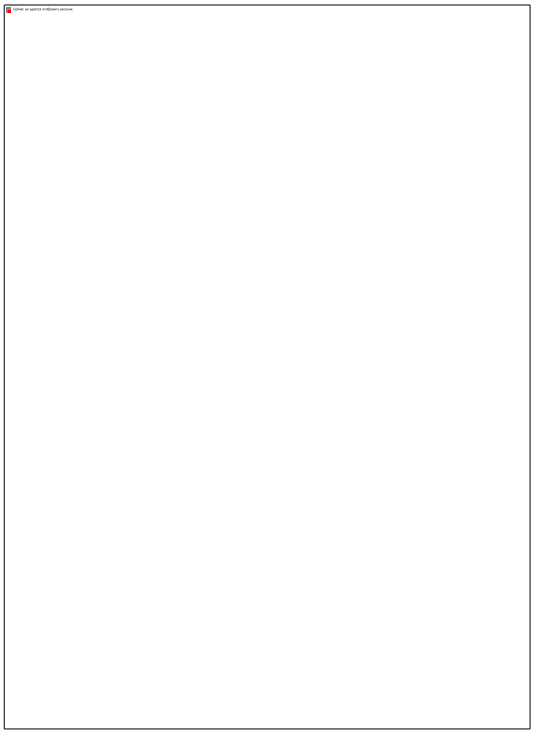
Если условие выполняется, то есть a>b, то следующим выполняется действие по стрелке «Да». Если условие не выполняется, то осуществляется переход по стрелке «Нет».

- блок Модификация используется для организации циклических (повторяющихся) действий.



- блок Предопределенный процесс используется для указания обращений к ранее созданным алгоритмам и программам, в том числе и библиотечным подпрограммам.
- блок Ввод-Вывод. При решении задачи на компьютере ввод исходных данных может осуществляться различными способами, например, с клавиатуры, с жесткого диска, с флэш-карты т. д. Задание численных значений исходных данных называется вводом, а отображение результатов расчета на экране монитора или с помощью принтера на бумаге – выводом. Если ввод-вывод не привязан к конкретному устройству, то обозначается параллелограммом. Если необходимо указать конкретное устройство ввода или вывода, то используются специальные геометрические фигуры.

В качестве примера графического способа описания алгоритмов с помощью блок-схем запишем алгоритм нахождения площади прямоугольника:



Внутри каждого блока записывается соответствующее действие. Последовательность выполнения задается соединительной линией со стрелочкой.

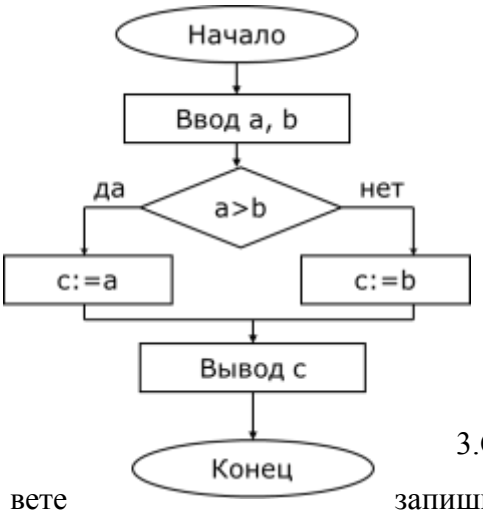
Последовательность выполнения сверху вниз и слева направо принята за основную.

Если в алгоритме не нарушается основная последовательность, то стрелочки можно не указывать. В остальных случаях последовательность выполнения блоков обозначается стрелочкой обязательно. В нашем примере основная последовательность выполнения – сверху вниз.

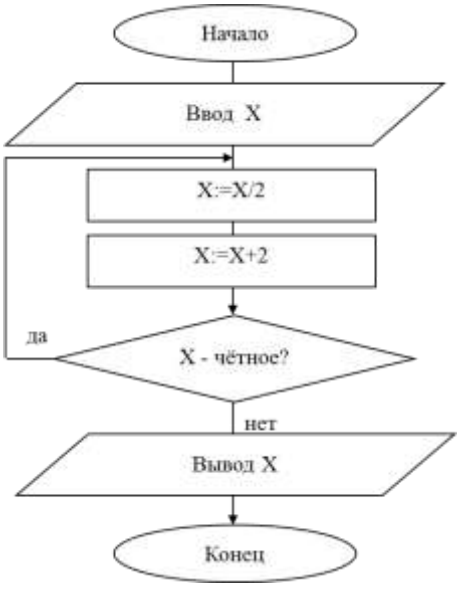
Задания:

1. Чему будет равно значение переменной s, если a=5, b=7?

2.Чему будет равно значение переменной c, если a=5, b=7?



3.Сколько раз выполнится тело цикла для числа 52? В ответе запишите число повторов.



Кейс № 3 «Основные понятия языка Паскаль»

Алфавит языка - это конечный набор символов, которые используются при написании любой конструкции на языке.

Алфавит языка Паскаль можно разбить на 3 группы:

a. буквы строчные и прописные буквы латинского и русского алфавита;
b. цифры - арабские 0..9
c. специальные символы - (+,-,*,/,=,,,=,.:;{}' := ( )), служебные слова: and, begin, const, div, do, else, for, и т.д.

Переменная - это область памяти, названная собственным именем, которая может менять свое значение в процессе выполнения программы. Переменная характеризуется именем, типом и значением.

Константа - не изменяет своего значения в процессе выполнения программы, она может быть задана явно своим значением или обозначена именем.

Имя (идентификатор) переменной или константы задается латинскими буквами и арабскими цифрами. В качестве идентификатора нельзя использовать служебные слова. Идентификатор должен быть уникальным, т.е. в данном блоке программы один идентификатор не может быть использован для обозначения более чем одного объекта.

Например: Zap, gor,X, p1, summa, a28,rar_1, proba, x1,y1,max,min ит.д.

Типы данных

В Паскале существуют следующие стандартные типы данных:

- целочисленный (integer);
- вещественный (real);
- логический (boolean);
- символьный (char);
- строка (string).

Итак, если переменная принадлежит типу integer, то она принимает только целочисленные значения, если принадлежит логическому типу boolean, то принимает значения либо true, либо false и т.д.

В отличие от языка Паскаль, где определен один целый тип Integer и один вещественный тип Real, в TurboPascal имеется 5 стандартных целых типов и 5 стандартных вещественных типов. Их рассмотрим позже.

Из вышеперечисленных типов целые типы – логический и символьный - относятся к порядковым типам, т.е. каждый из них имеет конечное число возможных значений.

Вещественные типы тоже имеют конечное число значений, но количество возможных значений настолько велико, что сопоставить с каждым из них целое число (его номер) невозможно.

Порядковые и вещественные типы относятся к простым типам.

Выражения

Арифметические выражения строятся из чисел и переменных с помощью знаков арифметических операций:

Сложение +, вычитание -, умножение *, деление/, целочисленное деление div, вычисление остатка от целочисленного деления mod.

Div – это целочисленное деление, вычисляющее частное от деления двух натуральных чисел.

Пример: 6:4=1 (остаток 2);

6 div 4=1 (целое частное);

Mod – это вычисление остатка от целочисленного деления

Пример: 6 mod 4=2 (остаток).

В выражениях могут быть использованы круглые скобки и некоторые функции. Арифметические функции можно использовать только с величинами целого и вещественного типов (таблица 1)

таблица 1

Арифметические функции

Математическая запись	Запись на Паскале	Тип результата
$\|x\|$	abs(x)	совпадает с типом x
x^2	sqr(x)	--------//---------
$\sqrt{x}$	sqrt(x)	вещественный
sin x	sin(x)	--------//---------
cos x	cos(x)	--------//---------
arctg x	arctan(x)	--------//---------
e^x	exp(x)	--------//---------
ln x	ln(x)	--------//---------
дробная часть числа	frac(x)	--------//---------
целая часть числа	int(x)	--------//---------
π=3.1415926535	pi	--------//---------
округление вещественного числа до ближайшего целого	round(x)	целый
получение целой части вещественного числа	trunc(x)	--------//---------

Для возведения в произвольную степень используется формула:

обобщенная показательно-степенная функция

$x^8 = e^{8 \ln x}$, на Паскале запишется так: exp(8*ln(x)).

Вычисление значений выражений выполняется в определенном порядок. Все действия выполняются в соответствии с их приоритетом:

1. вычисления в круглых скобках;
2. вычисления значений функций;
3. *, /, div, mod;
4. +, -.

Выражения на Паскале записываются в одну строку. Для указания правильного порядка действий используются круглые скобки.

Пример: записать на Паскале арифметические выражения:

1. $\frac{6\cos(x) - x^2}{\sqrt{2\sin(x)}}$ 2. $\frac{6y^3}{\ln(x)}$

Итак, на Паскале запишутся они так:

1. (6*cos(x) *sqr(x))/sqrt(2*sin(x));
2. (6*exp(3*ln(y)))/ln(x);

Первая программа на языке Pascal. Линейный алгоритм

Программа на языке Pascal состоит из двух разделов – раздела описаний и раздела операторов или, как часто выражаются программисты, главной программы. В разделе описаний мы сообщаем компьютеру о том, какие данные нам понадобятся для выполнения программы и другую необходимую информацию. Данные, которые меняются в процессе выполнения программы, называются переменными. Например, мы хотим написать программу, которая вычисляет площадь и периметр квадрата. Очевидно, что для решения этой задачи нам необходимы три переменные: для стороны квадрата, для площади и для периметра. Назовем их соответственно a, s и p.

Далее: мы должны определить, какого типа это будут переменные. В языке Pascal есть несколько типов переменных, с которыми мы познакомимся в ходе обучения. В данном случае мы имеем дело с вещественными переменными, что и должны отразить в нашей программе. Связано это вот с чем. Когда мы описываем переменную, компьютер отводит под нее память. Понятно, что для, например, целой и вещественной переменной требуется разная структура памяти. Для целого числа не нужно отводить память под десятичный разделитель (в языке Pascal это точка, а не запятая, к которой мы привыкли) и для знаков «после запятой». Вещественные переменные в языке Pascalописываются словом real.

Приступим теперь к созданию текста программы вычисления площади и периметра квадрата. В ранних версиях языка Pascal текст программы должен начинаться со слова program и имени программы. Например:

Programkvadrat;

Седьмая версия не требует обязательно начинать программу со слова program. В ней можно сразу начинать программу с раздела описаний.

Описание переменных начинается со слова var (от англ. Variable – переменная)

Итак, набираем в окне редактор текст:

Vara,s,p: real;

После слова var мы должны поместить хотя бы один список переменных. Переменные в списке перечисляются через запятую. Перед их типом ставится двоеточие. Точка с запятой является разделителем, означающим окончание списка. В тексте главной программы точка с запятой означает окончание той или иной операции

Обозначения a,s,p в языках программирования называются идентификаторами. Язык Pascal не различает заглавные или строчные буквы: a и A для него один и тот же идентификатор.

Есть три правила, по которым создаются идентификаторы в языке Pascal:

1. Идентификатор может содержать только буквы и цифры.
2. Идентификатор начинается только с буквы (a2 – правильно, 2a – неправильно)
3. Нельзя использовать в качестве идентификатора зарезервированные слова из языка Pascal (например, begin, end, var и т.д.).

Можно использовать в идентификаторе подчерк. Например, можно ввести такой идентификатор: my_program. Нельзя использовать один и тот же идентификатор дважды для обозначения двух разных переменных. Язык Pascal не чувствителен к регистру: он не различает прописные и строчные буквы. Рекомендуем создавать осмысленные идентификаторы. Так легче в большой программе вспомнить их значение.

После этих предварительных замечаний мы можем продолжить писать вашу программу. Переходим теперь к разделу операторов.

Главная программа начинается со слова *begin* и заканчивается словом *end.* В конце главной программы ставится точка. Между этими словами расположены команды- операторы, которые будут выполняться при исполнении программы. В целом программа должна выглядеть так:

{*раздел описаний}*

Var

a,s,p:real;

{главная программа}

begin

{операторы}

end. {точка после end обязательна в конце программы}

Пришло время разработать алгоритм нашей программы. Ее назначение мы определили: она будет вычислять площадь и периметр квадрата. Какие же действия и в какой последовательности мы должны выполнить?

Наша программа ориентирована на некоторого пользователя, который должен ввести в нее исходные данные – сторону квадрата. Следовательно, пользователя нужно пригласить сделать это, для чего следует вывести на экран соответствующий текст. Например: «Введите длину стороны квадрата». Вывод на экран осуществляется в специальном пользовательском окне. Вы можете увидеть его, набрав комбинацию клавиш AIt+F5. Далее, необходимо ввести в компьютер длину стороны квадрата. Потом проводятся вычисления и, наконец, нужно вывести на экран значения площади и периметра квадрата.

Эта последовательность действий не зависит от значений входных данных. Такой алгоритм называется в программировании линейным.

Приведем теперь полный текст программы и поясним значение каждого оператора. Вам следует набрать этот текст на своем компьютере. Текст в фигурных скобках называется комментариями. Его видит программист, но не воспринимает компьютер. Программисту рекомендуется писать развернутые комментарии к своей программе. Это поможет при отладке и доработке ее текста. Но сейчас для краткости вы можете их не печатать, особенно если вы не вполне освоились с клавиатурой вашего PC. Вам понадобится использовать русский шрифт. На вашем компьютере должен быть установлен русификатор, с помощью которого вы переключите клавиатуру на русский шрифт.

Итак:

{*раздел описаний}*

Var

a,s,p:real;

{главная программа}

begin

writeln(‘Введите длину стороны квадрата’); *{Печать приглашения}*

readln(a); *{Ввод стороны квадрата}*

s:=a*a; *{Вычисление площади}*

p:=4*a; *{Вычисление периметра}*

writeln (‘Площадь квадрата =‘,s:7:3); *{Вывод значения площади}*

writeln (‘Периметр квадрата =‘,s:7:3); *{Вывод значения периметра}*

readln; {Остановка до нажатия}

end. {точка после обязательна в конце программы}

Обращаем ваше внимание на то, что операторы, расположенные между begin и end сдвинуты вправо. Это называется структурированием программы. В вашей сложной программе будет много операторных скобок begin – end. Найти эту ошибку в неструктурированной программе непросто. Поэтому советуем с самого начала привыкать писать программу структурно.

Оператор writeln (‘Введите длину стороны квадрата’); выводит на экран сообщение, которое необходимо взять в апострофы. На экране появляется соответствующая надпись. Оператор readln (a) — это оператор считывания. После того, как вы напечатаете на экране значение стороны квадрата и нажмете клавишу Enter он считает это значение и поместит его в память компьютера, отведенную под переменную a.

Далее следуют вычисления. Над вещественными числами в языке Pascal определены четыре арифметических действия

\+ - сложение;

\- - вычитание;

* - умножение;

/ - деление.

Оператор:= называется оператором присваивания. Он вычисляет (в поле операций) значение выражения, представленного в правой части оператора, и записывает его в память, отведенную под переменную, указанную в левой части. Таким образом, переменные s и p принимают значения соответственно площади и периметра квадрата.

Далее: знакомый нам уже оператор writeln выводит па экран надпись и значение переменной s. Переменная отделяется от надписи запятой. Выражение s:7:3 обозначает, что для вывода значения s отведено 7 знаков, в том числе 3 «после точки».т.е. под дробную часть числа. Это называется «форматом числа». Приведенный формат не единственный, могут быть и другие варианты. Если этого не сделать, компьютер выдает ответ в неудобной для пользователя экспоненциальной форме (когда программа у вас заработает, попробуйте написать вместо s:7:3 просто s.Посмотрите, что у вас получится).

Вывод на экран значения периметра делается таким же образом. Далее, мы видим в программе оператор readln. Если этот оператор не имеет скобок, то он выполняет следующую функцию: останавливает программу до нажатия клавиши Enter. Если этого не сделать, то после выполнения программы пользовательское окно закроется, и мы увидим перед собой окно отладчика-редактора с текстом программы. Результаты вычислений можно

будет увидеть, нажав комбинацию клавиш A1t-F5. Это, однако, неудобно. Оператор readln позволяет нам видеть результаты вычислений до тех пор, пока мы не нажмем Enter.

Теперь, когда текст программы, готов нам необходимо его сохранить. Для этого нажмите клавишу F2. Появится диалоговое окно SaveAs (сохранить как). Введите в него имя. Например, kvadrat. Расширение pas будет автоматически присвоено файлу (помеченной области на диске компьютера), в котором сохранится текст. В верхней части окна редактора появится имя файла: kvadrat. pas. Отметим, что приповтором нажатии клавиши F2 файл сохранится под тем же именем. Советуем чаще сохранять файл после внесения в него изменений. Предыдущая версия файла также будет сохранена в файле с расширением bak. Чтобы сохранить файл под новым именем, необходимо войти в меню File, и выбрать пункт SaveAs.

Теперь, после того, как программа сохранена, необходимо провести ее компиляцию. Нажмите для этого клавишу F9 или Alt+F9. Специальная программа – компилятор – переработает ваши команды в коды, понятные компьютеру и сохранит их в исполняемом файле с расширением exe. Если в тексте программы есть ошибки, компиляция не будет завершена и вам будет указано на место в программе, где находится ошибка. Курсор остановится под строкой, в которой находится ошибка, и на экране появится красная полоса, в которой будет указан характер ошибки. После исправления ошибки повторите компиляцию если в программе есть еще ошибки, компилятор найдет их. В противном случае перед вами появится диалоговое окно, в котором будет написано «Compilesuccessful. Pressanykey». Нажмите после этого любую клавишу.

Наиболее часто встречающаяся у начинающих ошибка – «Unknownidentifier» (неизвестный идентификатор). Как правило, она возникает по двум причинам: либо переменная не указана в списке переменных после слова var, либо просто неправильно написан какой-нибудь идентификатор. Но могут быть и другие ошибки. Если вам непонятен смысл сообщения в красной полосе, нажмите клавишу F1. Вы получите развернутую подсказку. Умение исправлять свои ошибки пригодится вам в программировании, и в жизни!

Теперь можно запускать программу и продолжать ее отладку. Компилятор находит ошибки только в программировании, ошибки в алгоритме программист должен найти сам. Например, если вы вместо знака умножения поставите знак сложения, то сообщения об ошибке не появится, но программа будет работать неправильно. Для ее проверки запускают какой -нибудь тест, результат которого вам известен заранее. В данном случае се, конечно, очень просто. Примите сторону квадрата равной, скажем, двум. Если в результате вычислений вы получите 4 и 8, значит, ваша программа работает правильно.

Итак, запускаем программу. Для этого нажмите комбинацию клавиш Ctrl+F9. Появится приглашение. Наберите на клавиатуре значение стороны квадрата. Нажмите Enter. Посмотрите на результаты вычислений. Нажмите опять Enter и вернитесь в окно редактора к тексту программы.

Если все у вас получилось, примите мои поздравления: вы написали и отладили первую в своей жизни программу.

Теперь нужно выйти из среды TurboPascal. Для этого нужно либо выбрать в File пункт Exit, либо нажать комбинацию клавиш Alt+X. Возможно при этом появится предложение сохранить программу. Выполните соответствующие действия по сохранению.

Когда выйдете из TurboPascal, откройте папку, в которой вы сохраняете свои файлы. Там вы должны найти файлы kvadrat.pas и kvadrat.exe. щелкните мышью по exe файлу. При этом заработает ваша программа вычисления площади и периметра квадрата. В дальнейшем вы сможете использовать исполняемый файл своей игры «отдельно» от системы TurboPascal.

Итак, мы завершили создание нашей первой программы. Напишем теперь вторую: вычисление площади и периметра прямоугольника. Основная наша задача при этом, научиться использовать свои собственные наработки. Для этого нужно уметь модифицировать текст программы и сохранять его под другим именем. Общий совет: старайтесь писать, как можно меньше. Используйте переименование, копирование, перенос текста из окна в окно. Всем этим приемам мы вас постепенно научим.

Приступаем. Войдите в TurboPascal. Откройте созданный вами файл kvadrat.pas. Для этого нажмите клавишу F3 или выберите в меню File пункт Open. Появится диалоговое окно, в котором вы увидите список файлов с расширением pas. Войти в этот список можно с помощью клавиши Tab. По списку перемещаемся стрелками, выбрав нужный файл нажимаем Enter.

После открытия файла, начинаем его модифицировать. Для одной из сторон прямоугольника мы по-прежнему будем использовать переменную a. Но нам понадобится еще одна переменная для другой стороны прямоугольника. Назовем ее b и включим в список вещественных переменных. Оператор чтения переменных приобретет вид: readln (a,b). При этом при запуске программы значение переменных можно вводить двумя способами: либо набирать их по очереди, каждый раз нажимая Enter, либо набрать их через пробел, а потом нажать один раз Enter.

Изменятся также формулы для вычисления периметра и площади, и в тексте приглашения и вывода результатов вместо слова «квадрат» появится слово «прямоугольник».

Когда вы проделаете все эти операции, получится следующий текст (комментарии для краткости не приводятся):

```
Var
a,b,s,p:real;
begin
writeln( ‘Введите длину сторон прямоугольника’);
readln(a,b);
s:=a*b;
p:=2*(a+b);
writeln(‘Площадь прямоугольника=’,s:7:3);
writeln(‘Периметр прямоугольника=’,p:7:3);
readln;
end.
```

Выберите теперь в меню File пункт SaveAs. Сохраните файл под именем pram.pas. Вот и все. Теперь откомпилируйте программу, исправьте ошибки и приступайте к запуску и отладке.

Задания:

1..Вычислить

24/(3*4)-24/3/4+24/3*4;

2.Записать на Паскале арифметические выражения:

$$\frac{5y^6}{\sqrt{x}+\cos(y-3x^2)}; \quad \frac{\left|\cos(x-5y^2)\right|}{\sqrt{6\ln(y)}}$$

3. Можно ли утверждать, что значение выражения (1/6)*6-1 равно нулю?

4. Почему параметрами процедуры ввода read могут быть только переменные, а не числа или выражения, скажем, вида X+1?

5.Если в качестве исходных данных задано пять чисел, то можно ли ввести пятое из них, не введя первые четыре?

Приложение 3

Описание форм организации совместной деятельности учащихся на уроке

Урок-игра. Пример: тематическое повторение или в игровой форме расширение кругозора детей.

Учебное задание-проект.Выполнение совместных проектов в группах; подготовка текстовых файлов; создание документов с изображением, таблицами и другими графически-

ми объектами; обсуждение правок в документах с другими соавторамина основе облачных вычислений, применяемыми в образовательном процессе, являются GoogleAppsEducationEdition.

Найди ошибку. Детям выдаются карточки со словами, изученными по какой-либо теме, в них допущены ошибки. Нужно их найти, исправить и объяснить.

Игра «Продолжи». Основана на выполнении заданий разного рода группой «по цепочке». Учащиеся обобщают и систематизируют изученный материал.

Снежный ком. Работа в группе, которая начинается с решения индивидуального задания. Все учащиеся получают аналогичные задания и самостоятельно выполняют их. После этого следует работа в парах. В парах учащиеся предлагают свои способы решения данного задания, из которых выбирается лучшее. Далее две пары объединяются, и работа продолжается в группе из четырех человек, где снова происходит обсуждение решений и выбирается лучшее из них. В конце работы все учащиеся попадают в одну группу. На этом последнем этапе уже не происходит обсуждения решений, группы делают доклады о своей работе.

Пазлы. Учитель делит тему на несколько частей так, чтобы каждая группа получила бы свою часть темы. Также все группы получают список необходимых источников или сами учебные материалы, с помощью которых они изучают основы предложенной части темы.

После изучения материала или выполнения задания группы переформируются так, чтобы в каждую новую группу попали по 1 человеку от каждой прежней группы.

Каждый член новой группы объясняет своим новым коллегам свою часть темы, основы которой он изучил в составе предыдущей группы и отвечает на заданные вопросы.

В заключение работы делают выводы.

Ключевые термины. Учитель выбирает из текста 4-5 ключевых слов и выписывает их на доску.

Вариант «а». Парам отводится 5 минут на то, чтобы методом мозговой атаки дать общую трактовку этих терминов и предположить, как они будут фигурировать в последующем тексте.

Вариант «б». Учащимся предлагается в группе или индивидуально составить и записать свою версию рассказа, употребив все предложенные ключевые термины.

При знакомстве с исходным содержанием, учащиеся сопоставляют «свою» версия и версию «оригинального текста». Описанное задание обычно используется на стадии «вызова» при обобщении имеющихся у ученика знаний, однако на стадии «рефлексии», обобщения и осмысления полученной информации, целесообразно вернуться к ключевым терминам и обсудить обнаруженные совпадения и выявленные разногласия. Использование

данной формы развивает воображение, фантазию, способствует активизации внимания при знакомстве с текстом оригинала.

Прием «Зигзаг». Или метод пилы. Учащиеся организуются в группы по 4-5 человек для работы над учебным материалом, который разбит на фрагменты.

Затем ребята, изучающие один и тот же вопрос, но состоящие в разных группах, встречаются и обмениваются информацией как эксперты по данному вопросу. Это называется «встречей экспертов». Затем они возвращаются в свои группы и обучают всему новому, что узнали сами, других членов группы. Те, в свою очередь, докладывают о своей части задания (как зубцы одной пилы).