ГБОУ РК «Феодосийская санаторная школа-интернат»

План-конспект уроков
по Информатике и ИКТ в 9 классе

Разработал учитель:
Калачиков Александр Владимирович

г. Феодосия

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 1
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Техника безопасности и организация рабочего места. Управление и кибернетика. Управление с обратной связью.
Тип урока: Урок-лекция.
Цели урока:
Образовательные
Изучение вопросов:
- Кибернетическая модель управления
- Управление без обратной связи и с обратной связью
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:
1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Вы уже знакомы с некоторыми областями использования компьютеров. Знаете, что с помощью компьютера можно печатать книги, выполнять чертежи и рисунки; быстро передавать информацию на большие расстояния, создавать компьютерные справочники на любую тему; производить расчеты. Существует еще одно важное приложение компьютерной техники — использование компьютеров для управления.

Возникновение кибернетики

В 1948 году в США и Европе вышла книга американского математика *Норберта Винера* «Кибернетика, или управление и связь в животном и машине».

Эта книга провозгласила рождение новой науки — кибернетики.

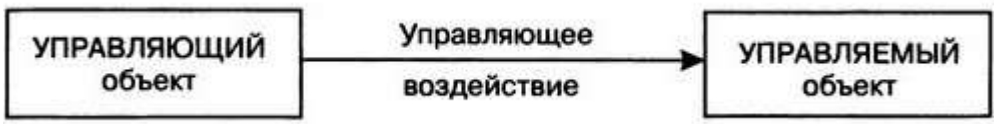
Не случайно время появления этого научного направления совпало с созданием первых ЭВМ.

Н. Винер предвидел, что использование ЭВМ для управления станет одним из важнейших их приложений, а для этого потребуется глубокий теоретический анализ самого процесса управления. Этому и посвящена наука кибернетика.

Что такое управление

Управление есть целенаправленное воздействие одних объектов, которые являются управляющими, на другие объекты — управляемые.

Простейшая ситуация — два объекта: один — управляющий, второй — управляемый. Например: человек и телевизор, хозяин и собака, светофор и автомобиль. В первом приближении взаимодействие между такими объектами можно описать схемой, изображенной на рисунке ниже:



В приведенных примерах управляющее воздействие производится в разных формах: человек нажимает кнопку или поворачивает ручку управления телевизором; хозяин голосом подает команду собаке; светофор разными цветами управляет движением автомобилей и пешеходов на перекрестке.

С кибернетической точки зрения все варианты управляющих воздействий следует рассматривать как *управляющую информацию*, передаваемую в форме команд.

В примере с телевизором через пульт управления передаются команды следующего типа: «включить/выключить», «переключить канал», «увеличить/уменьшить громкость». Хозяин передает собаке команды голосом: «Сидеть!», «Лежать!», «Взять». Световые сигналы светофора шофер воспринимает как команды: красный — «стоять», желтый — «приготовиться», зеленый — «ехать».

Алгоритм управления

В данном выше определении сказано, что управление есть целенаправленный процесс, т. е. команды отдаются не случайным образом, а с вполне определенной целью. В простейшем случае цель может быть достигнута после выполнения одной команды. Для достижения более сложной цели бывает необходимо выполнить последовательность (серию) команд.

Последовательность команд по управлению объектом, выполнение которой приводит к достижению заранее поставленной цели, называется **алгоритмом управления**.

В таком случае объект управления можно назвать *исполнителем управляющего алгоритма*. Значит, в приведенных выше примерах телевизор, собака, автомобиль являются исполнителями управляющих алгоритмов, направленных на вполне конкретные цели (найти интересующую передачу, выполнить определенное задание хозяина, благополучно проехать перекресток).

С точки зрения кибернетики взаимодействие между управляющим и управляемым объектами рассматривается как *информационный процесс*. С этой позиции оказалось, что самые разнообразные процессы управления в природе, технике, обществе происходят сходным образом, подчиняются одним и тем же принципам.

Линейный алгоритм

Если внимательно обдумать рассмотренные в предыдущем параграфе примеры, то можно прийти к выводу, что строго в соответствии с рассмотренной ранее схемой работает только система «светофор – автомобили». Светофор «не глядя» управляет движением машин, не обращая внимания на обстановку на перекрестке. Вот алгоритм работы светофора:

КРАСНЫЙ — ЗЕЛЕНЫЙ — ЖЕЛТЫЙ — КРАСНЫЙ — ЗЕЛЕНЫЙ — ЖЕЛТЫЙ — КРАСНЫЙ и т. д.

Такой алгоритм называется ***линейным*** или ***последовательным***.

Обратная связь

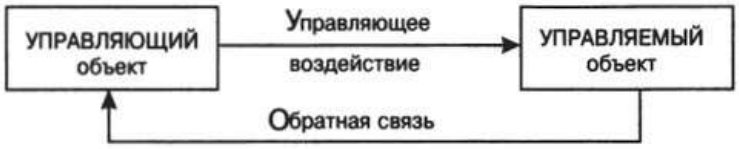
Совсем иначе протекает процесс управления телевизором или собакой. Прежде чем отдать очередную команду, человек смотрит на состояние объекта управления, на результат выполнения предыдущей команды. Если он не нашел нужную передачу на данном канале, то он переключит телевизор на следующий канал; если собака не выполнила команду «лежать!» хозяин повторит эту команду.

Из этих примеров можно сделать вывод, что управление происходит эффективнее, если управляющий не только отдает команды, т. е. работает *прямая связь*, но и принимает информацию от объекта управления о его состоянии. Этот процесс называется *обратной связью*.

Обратная связь — это процесс передачи информации о состоянии объекта управления управляющему объекту.

Модель управления с обратной связью

Управлению с обратной связью соответствует схема, изображенная на рисунке ниже:



Циклы и ветвлении в алгоритмах

Вот как можно записать алгоритм поиска нужной передачи по телевизору:

ВКЛЮЧИТЬ ТЕЛЕВИЗОР НА 1-М КАНАЛЕ
ПОКА НЕ БУДЕТ НАЙДЕНА ИСКОМАЯ ПЕРЕДАЧА,
ПОВТОРЯТЬ:
ПЕРЕКЛЮЧАТЬ ТЕЛЕВИЗОР НА СЛЕДУЮЩИЙ КАНАЛ

В этом алгоритме содержится указание на повторение одних и тех же действий (переключить канал) по некоторому условию (пока не найдем передачу). Такой алгоритм называется *циклическим*.

Если вместо светофора на перекрестке дорог работает милиционер регулировщик, то управление движением станет более рациональным. Регулировщик следит за скоплением машин на пересекающихся дорогах и дает «зеленую улицу» в том направлении, в котором в данный момент это нужнее. Нередко из-за «безмозглого» управления светофора на дорогах возникают «пробки». И тут на помощь может прийти регулировщик.

Назовем пересекающиеся дороги Дорога-1 и Дорога-2. Логика управления движением описывается следующим алгоритмом:

ЕСЛИ НА ДОРОГЕ-1 СКОПИЛОСЬ БОЛЬШЕ МАШИН
ТО ОТКРЫТЬ ДВИЖЕНИЕ ПО ДОРОГЕ-1
ИНАЧЕ ОТКРЫТЬ ДВИЖЕНИЕ ПО ДОРОГЕ-2

Здесь по определенному условию происходит выбор одного из двух действий. Такой алгоритм называется *ветвящимся*. Проверка выполнения условия и в первом, и во втором примере стала возможна благодаря обратной связи: телезритель наблюдает за состоянием телевизора, милиционер наблюдает за состоянием движения на дорогах.

Итак, в варианте управления без обратной связи алгоритм может представлять собой только однозадачную (линейную) последовательность команд. При наличии обратной связи и «интеллектуального» управляющего объекта алгоритмы управления могут иметь сложную структуру, содержащую альтернативные команды (ветвления) и повторяющиеся команды (циклы).

При наличии обратной связи алгоритм может быть более гибким, допускающим проверку условий, ветвления и циклы.

Принцип управления с обратной связью и есть основной закон, открытый наукой кибернетикой. Он действует в системах самой разной природы: технических, биологических, социальных.

Системы с программным управлением

Системы, в которых роль управляющего объекта поручается компьютеру, называются **автоматическими системами с программным управлением**.

Для функционирования такой системы,
во-первых, между компьютером и объектом управления должна быть обеспечена прямая и обратная связь,
во-вторых, в память компьютера должна быть заложена программа управления (алгоритм, записанный на языке программирования).

Поэтому такой способ управления называют *программным управлением*.

Программное управление широко используется в технических системах: автопилот в самолете, автоматическая линия на заводе, ускоритель элементарных частиц в физической лаборатории, атомный реактор на электростанции и пр.

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§1,2, вопросы к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!
Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 2
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Понятие алгоритма и его свойства. Исполнитель алгоритмов: назначение, среда, система команд, режимы работы.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов:
- Понятие алгоритма и его свойства
- Исполнитель алгоритмов
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Происхождение понятия «алгоритм»

Само слово «алгоритм» происходит от имени выдающегося математика средневекового Востока Мухаммеда ибн Мусы аль-Хорезми (787 - 850 г.). Им были предложены приемы выполнения арифметических вычислений с многозначными числами (вам они хорошо знакомы из школьной математики). Позже в Европе эти приемы назвали алгоритмами, от *Algorithmi* — латинского написания имени аль-Хорезми. В наше время понятие алгоритма понимается шире, не ограничивается только арифметическими вычислениями.

Исполнитель алгоритма

Из предыдущего параграфа вы узнали, что алгоритм — это последовательность команд управления каким-либо объектом. Мы назвали такой объект объектом управления или *исполнителем алгоритма*. Им может быть как техническое устройство, так и живое существо.

Рассмотрим исполнителя — человека. Для него можно сформулировать множество алгоритмов, например алгоритмы арифметических вычислений. С таким же успехом можно назвать алгоритмами множество различных инструкций, предписывающих последовательность действий человека для выполнения какой-либо работы. Например, кулинарный рецепт — это алгоритм работы повара с целью приготовления блюда; инструкция по сборке машинки из деталей детского конструктора — алгоритм для ребенка; инструкция по использованию кухонного комбайна — алгоритм для домохозяйки.

Вы, наверное, никогда не задумывались над тем, какое количество алгоритмов вам известно. Жизненный опыт человека растет с увеличением числа освоенных им алгоритмов. Например, чтобы ребенок научился покупать в магазине хлеб, ему нужно сначала рассказать (а лучше показать), как это делается. Освоив «алгоритм покупки хлеба», он в дальнейшем будет успешно выполнять эту работу.

Поиск выигрышной тактики, а следовательно, и алгоритма несложной игры — интересная и полезная задача. Рассмотрим одну из таких игр, которая называется игрой Баше.

Играют двое. Перед ними 21 предмет, допустим, камни (также может быть 11, 16, 26 и т. д.). Игроки берут камни по очереди. За один ход можно взять 1, 2, 3, 4 камня. Проигрывает тот, кто забирает последний камень.

Имеется выигрышная тактика для игрока, берущего камни вторым. Она заключается в том, чтобы брать такое количество камней, которое дополняет число камней, взятых соперником на предыдущем ходе, до пяти. Этот алгоритм можно описать в виде последовательности команд:

алг Игра Баше

нач

1. Предоставить ход сопернику.
2. Взять столько камней, чтобы в сумме с предыдущим ходом соперника получилось 5.
3. Если остался один камень, то объявить о своем выигрыше, иначе вернуться к выполнению команды 1.

кон

Игрок, строго следующий этому алгоритму, будет всегда выигрывать, даже если он не понимает, почему так происходит.

Алгоритмический язык

В приведенном примере записи алгоритма используется символика учебного ***Алгоритмического языка (АЯ)***.

Из примера видно, что при записи алгоритма на АЯ вначале пишется заголовок, начинающийся со ***служебного слова алг*** (сокращенное слово «алгоритм»). Затем указывается название алгоритма, которое составитель алгоритма придумывает сам. Следующая часть называется телом алгоритма. Она начинается со служебного слова нач (начало) и заканчивается словом ***кон*** (конец). Тело алгоритма представляет собой последовательность команд для исполнителя.

Здесь и в дальнейшем служебные слова в алгоритмах на алгоритмическом языке будут записываться жирным шрифтом. В языках программирования (как и в АЯ) служебными называются слова, которые всегда употребляются в одном и том же смысле.

Свойства алгоритма

Процесс решения задачи должен быть разбит на последовательность отдельно выполняемых шагов.

Это свойство алгоритма называется ***дискретностью***.

Всякий алгоритм составляется в расчете на конкретного исполнителя с учетом его возможностей. Для того чтобы алгоритм был выполним, нельзя включать в него команды, которые исполнитель не в состоянии выполнить. Нельзя повару поручать работу токаря, какая бы подробная инструкция ему ни давалась. У каждого исполнителя имеется свой перечень команд, которые он может исполнить. Такой перечень называется ***системой команд исполнителя алгоритмов (СКН)***.

Алгоритм, составленный для конкретного исполнителя, должен включать только те команды, которые входят в систему команд исполнителя.

Это свойство алгоритма называется ***понятностью***.

Алгоритм не должен быть рассчитан на принятие каких-либо самостоятельных решений исполнителем, не предусмотренных составителем алгоритма.

Каждая команда алгоритма должна определять однозначное действие исполнителя.

Это свойство алгоритма называется ***точностью***.

Исполнение алгоритма должно завершиться за конечное число шагов.

Это свойство алгоритма называется ***конечностью***.

Для успешного выполнения любой работы мало иметь ее алгоритм. Всегда требуются еще какие-то ***исходные данные***, с которыми будет работать исполнитель (продукты для приготовления блюда, детали для сбора технического устройства и т. п.). Исполнителю, решающему :математическую задачу, требуется исходная числовая информация. Задача всегда формулируется так: дана исходная информация, требуется получить какой-то результат. В математике вы привыкли в таком: виде записывать условия задач. Например:

Дано: катеты прямоугольного треугольника **a**= 3 см; **b** = 4 см.

Найти: гипотенузу **c**

Алгоритм: решения этой задачи можно представить в таком виде:

алг Гипотенуза

нач

1. Возвести **a** в квадрат.
2. Возвести **b** в квадрат.
3. Сложить результаты действий 1 и 2.
4. Вычислить квадратный корень результата действия 3 и принять его за значение **c**.

кон.

Каждую из этих команд может выполнить любой человек, знающий основы математики, следовательно, они входят в его систему команд.

Еще пример: для поиска номера телефона нужного вам человека исходными данными являются: фамилия, инициалы человека и телефонная книга (точнее, информация, заключенная в телефонную книгу). Однако этого может оказаться недостаточно. Например, вы ищете номер телефона Смирнова

А. И. и обнаруживаете, что в книге пять строк с фамилией «Смирнов А. И». Ваши исходные данные оказались неполными для точного решения задачи (вместо одного номера телефона вы получили пять). Оказалось, что нужно знать еще домашний адрес.

Набор: «Фамилия — инициалы — телефонный справочник – адрес» является полным набором данных в этой ситуации.

Только имея полный набор данных, можно точно решить задачу.

Если исходные данные неполные, то задачу, либо совсем нельзя решить (ничего нельзя узнать про гипотенузу по одному катету), либо получается неоднозначное решение (пять номеров телефонов).

В задачах управления физическими объектами (автомобиль, самолет, станок и т. п.) исходными данными является информация о состоянии объекта управления, об обстановке, его окружающей.

Определение алгоритма

Обобщая все сказанное, сформулируем определение алгоритма.

Алгоритм — понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.

Формальное исполнение алгоритма

Если алгоритм обладает перечисленными выше свойствами, то работа по нему будет производиться исполнителем ***формально*** (т. е. без всяких элементов творчества со стороны исполнителя). На этом основана работа программно управляемых исполнителей-автоматов, например промышленных роботов. Робот-манипулятор может выполнять работу токаря, если он умеет выполнять все операции токаря (включать станок, закреплять резец, перемещать резец, замерять изделие). От исполнителя не требуется понимания сущности алгоритма, он должен лишь точно выполнять команды, не нарушая их последовательности.

Что такое программа

А что такое программа? Отличается ли чем-то программа от алгоритма?

Программа — это алгоритм, записанный на языке исполнителя.

Иначе можно сказать так: алгоритм и программа не отличаются по содержанию, но могут отличаться по форме.

Для алгоритма строго не определяется форма его представления. Алгоритм можно изобразить графически, можно — словесно, можно — какими-нибудь специальными значками, попятными только его автору. Но программа должна быть записана на языке исполнителя.

Назначение и возможности графического исполнителя (ГРИС)

Учебные исполнители используются для обучения составлению управляющих алгоритмов.

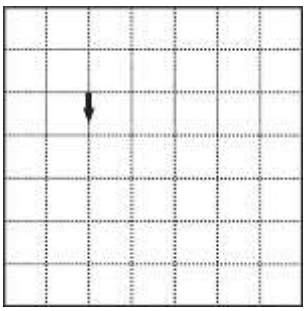
Есть много учебных исполнителей, придуманных для занятий по информатике. У них разные, часто забавные названия: Черепашка, Робот, Чертежник, Кенгуренок, Пылесосик, Муравей, Кукарача и другие. Одни исполнители создают рисунки на экране компьютера, другие складывают слова из кубиков с буквами, третьи перетаскивают предметы из одного места в другое. Все эти исполнители управляются программным путем. Любому из них свойственна определенная ***среда деятельности***, ***система команд управления***, ***режимы работы***.

В этой главе мы не будем детально описывать работу с каким-то реальным учебным исполнителем из вышеперечисленных (в компьютерных классах разных школ может быть разное программное обеспечение). Мы опишем условного исполнителя, который очень похож на некоторых из существующих в главном: системой команд, языком и приемами программирования.

Многие из учебных исполнителей занимаются рисованием на экране компьютера. Из названных выше это Черепашка, Кенгуренок, Чертежник. Эту группу можно назвать графическими исполнителями. Пусть наш гипотетический (придуманный) исполнитель тоже будет из этой «компании». Назовем его ГРИС, что значит «ГРафический ИСполнитель».

Что умеет делать ГРИС? Он может перемещаться по полю и своим хвостом рисовать на этом поле (предположим, что у него есть хвост, к которому привязан кусочек мела).

Обстановка, в которой действует исполнитель, называется ***средой исполнителя***. Среда графического исполнителя показана на рисунке ниже. Это лист (страница экрана) для рисования. ГРИС может перемещаться в горизонтальном и вертикальном направлениях с постоянным шагом. На рисунке пунктиром показана сетка с шагом, равным шагу исполнителя. Исполнитель может двигаться только по линиям этой сетки. ГРИС не может выходить за границы поля.



Состояние исполнителя на поле определяется, во-первых, его местоположением (в какой точке поля он находится), во-вторых, направлением (куда он смотрит). Направление будем определять, как на географической карте: вверх — на север, вниз — на юг, влево — на запад, вправо – на восток. ГРИС может шагать или прыгать по линиям сетки, а также поворачиваться. Поворачиваться он умеет только против часовой стрелки.

Графический исполнитель — это объект управления. А управлять им будем мы с вами. Целью управления является получение определенного рисунка. Понятно, что этот рисунок может состоять только из горизонтальных и вертикальных отрезков, в других направлениях ГРИС двигаться не умеет.

Задача обычно ставится так: исполнитель находится в данной точке поля, смотрит в данном направлении. Требуется получить определенный рисунок. Например: ГРИС находится в середине поля и смотрит на восток. Надо нарисовать букву «Т» с длиной каждой линии, равной четырем шагам.

Первоначально исполнителю придается исходное состояние. Это делается в специальном ***режиме установки***.

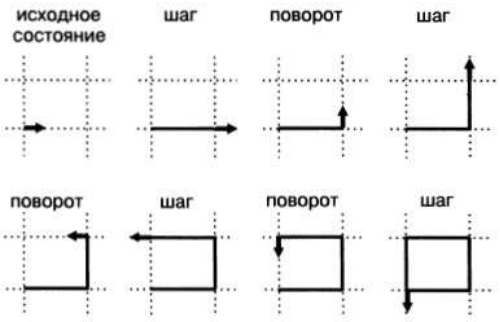
Теперь перейдем к управлению графическим исполнителем. Здесь возможны два режима: ***режим прямого управления*** и ***режим программного управления***.

Простые команды ГРИС

Работа в режиме прямого управления происходит так: человек отдает команду, ГРИС ее выполняет; затем отдается следующая команда и т. д. (как в примере с хозяином и собакой).

В режиме прямого управления система команд исполнителя следующая:

- *шаг* — перемещение ГРИС на один шаг вперед с рисованием линии;
- *поворот* — поворот на 90° против часовой стрелки;
- *прыжок* — перемещение на один шаг вперед без рисования линии.



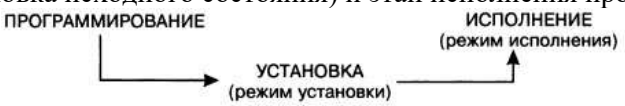
Эти команды будем называть *простыми командами*.

Например, пусть требуется нарисовать квадрат со стороной, равной одному шагу. Исходное положение ГРИС – в левом нижнем углу квадрата, направление — на восток. Будем отмечать состояние исполнителя маленькой стрелкой. Тогда последовательность команд и результаты их выполнения будут следующими:

Работа в программном режиме

Работа в программном режиме имитирует автоматическое управление исполнителем. Управляющая система (компьютер) обладает памятью, в которую заносится программа. Человек составляет программу и вводит ее в память. Затем ГРИС переводится в режим установки и человек вручную (с помощью определенных клавиш) устанавливает исходное состояние исполнителя. После этого производится переход в *режим исполнения* и ГРИС начинает работать по программе. Если возникает ситуация, при которой он не может выполнить очередную команду (выход за границу поля), то выполнение программы завершается аварийно. Если аварии не происходит, то работа исполнителя заканчивается на последней команде.

Таким образом, программное управление графическим исполнителем проходит этап подготовки (программирование и установка исходного состояния) и этап исполнения программы.



В режиме программного управления по-прежнему используются команды шаг, поворот, прыжок. Однако в этом режиме есть еще и другие команды. С ними вы познакомитесь позже.

Язык программирования для графического исполнителя — это учебный Алгоритмический язык (АЯ). Поэтому алгоритмы управления ГРИСом, записанные на АЯ, являются для него одновременно и программами.

Линейные программы для ГРИС

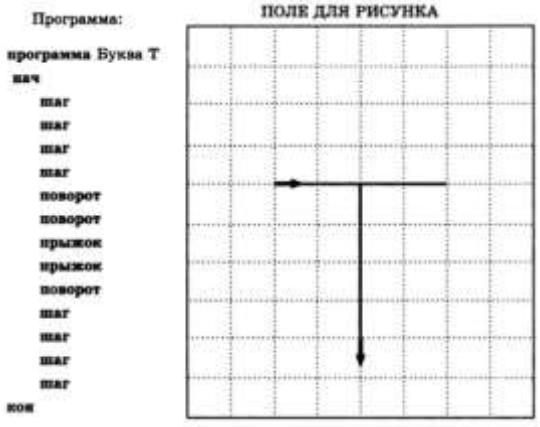
Будем осваивать программирование на примерах решения конкретных задач. С новыми командами СКИ будем знакомиться по мере появления потребности в них.

Задача 1. Составим и выполним программу, по которой ГРИС нарисует на поле букву «Т». Пусть длина вертикального и горизонтального отрезков должна быть равна четырем шагам.

Исходное состояние — чистый лист. Исполнитель находится в точке, где будет находиться левый конец горизонтального отрезка, направление — на восток.

Результат выполнения программы показан ниже.

Структура такой программы (алгоритма) называется линейной. Команды выполняются одна за другой, каждая только один раз.



Для решения этой задачи оказалось достаточно той части СКИ, которая используется в режиме прямого управления.

Коротко о главном:

- ГРИС — это графический исполнитель, назначение которого — получение чертежей, рисунков на экране дисплея.
- Управление ГРИС может происходить в режиме прямого управления или в режиме программного управления.
- С помощью команд шаг, поворот, прыжок в пределах рабочего поля можно построить любой рисунок, состоящий из вертикальных и горизонтальных отрезков. Структура управляющего алгоритма при этом будет линейной.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§3,4, вопросы к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 3
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №1 Работа с учебным исполнителем алгоритмов: построение линейных алгоритмов.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: построение линейных алгоритмов
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практической работы (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практической работы

Ход работы

1. Запускаем программу **Кумир**
2. В меню **Практикум – Загрузить курс** загружаем практический курс **practicum.kurs**
3. На вопрос **Создать тетрадь?** отвечаем **НЕТ**
4. Выполняем задачи **№1** и **№2** из подразделов **Знакомство** и **Линейные алгоритмы**
5. Дополнительное задание: выполняем оставшиеся задания из подразделов **Знакомство** и **Линейные алгоритмы**
6. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §4, задания №6,7 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!
Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 4
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Вспомогательные алгоритмы. Метод последовательной детализации и сборочный метод.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: обращение к вспомогательному алгоритму (процедуре), описание вспомогательного алгоритма (процедуры)
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

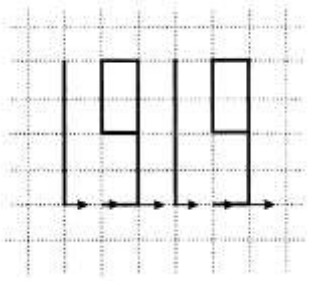
Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Что такое вспомогательный алгоритм
Решим следующую задачу:
Пусть требуется составить программу, по которой ГРИС напишет на экране четырехзначное число 1919.



Конечно, можно поступить так, как в предыдущей задаче, написав одну длинную программу, по которой исполнитель шаг за шагом нарисует эти цифры. Но с очевидностью возникает другая идея: поскольку здесь дважды повторяются цифры 1 и 9, нельзя ли сократить работу, написав программы рисования той и другой цифры только один раз? Это действительно можно сделать.

Алгоритм, по которому решается некоторая подзадача из основной задачи и который, как правило, выполняется многократно, называется **вспомогательным алгоритмом**.

Вспомогательный алгоритм, записанный на языке программирования, называется ***подпрограммой*** или ***процедурой***.

Обращение к вспомогательному алгоритму (процедуре)
В таком случае программа решения поставленной задачи разделяется на основную программу (основной алгоритм) и процедуры (вспомогательные алгоритмы). Каждая процедура должна иметь свое уникальное имя. Для рассматриваемой задачи имена процедур выберем следующими: ЕДИНИЦА и ДЕВЯТЬ. Тогда в основной программе команды обращения к этим процедурам будут такими:
сделай ЕДИНИЦА
сделай ДЕВЯТЬ
По этим командам управление передается соответствующим процедурам, и после их выполнения управление вернется к следующей команде основной программы.
Договоримся, что начальное и конечное состояния ГРИС при вычерчивании каждой цифры будут такими, как показано стрелками на рисунке выше (внизу, на восток). У единицы начальное и конечное состояния совпадают. Основная программа:
программа Число 1919
нач
сделай ЕДИНИЦА
прыжок
сделай ДЕВЯТЬ

прыжок
сделай ЕДИНИЦА
прыжок
сделай ДЕВЯТЬ
кон

Данный пример познакомил вас с новой командой из СКИ графического исполнителя — командой ***обращения к процедуре***. Ее формат, т. е. общий вид, следующий:

сделай <имя процедуры>

Описание вспомогательного алгоритма (процедуры)

Вот и все! Так просто! Но теперь надо «объяснить» исполнителю, что такое ЕДИНИЦА и что такое ДЕВЯТЬ. Это делается в ***описаниях процедур*** (здесь порядок выполнения — по столбцам):

```
процедура ЕДИНИЦА
нач
   поворот
   шаг
   шаг
   шаг
   шаг
   поворот
   поворот
   прыжок
   прыжок
   прыжок
   прыжок
   поворот
кон
```

```
процедура ДЕВЯТЬ
нач
   шаг
   поворот
   шаг
   шаг
   шаг
   шаг
   поворот
   шаг
   поворот
   шаг
   поворот
   шаг
   шаг
   поворот
   шаг
```

```
   поворот
   поворот
   поворот
   прыжок
   прыжок
   поворот
кон
```

Формат описания процедуры:

процедура <имя процедуры>

нач
<тело процедуры>
Кон

Имя в описании и имя в обращении должны точно совпадать (никаких склонений по падежам!). Описание процедур располагается после основной программы.

Добавив к программе описание процедуры, мы тем самым расширили систему команд исполнителя. В данной программе стало возможным использование команды обращения к этой процедуре.

Метод последовательной детализации

Использованный нами подход облегчает программирование сложных задач. Задача разбивается на более простые подзадачи. Решение каждой оформляется в виде вспомогательного алгоритма, а основной алгоритм организует связку между ними.

Метод программирования, при котором сначала пишется основная программа, в ней записываются обращения к пока еще не составленным подпрограммам, а потом описываются эти подпрограммы, называется ***методом последовательной (пошаговой) детализации***. Причем количество шагов детализации может быть гораздо большим, чем в нашем примере, поскольку сами подпрограммы могут содержать внутри себя обращения к другим подпрограммам.

Сборочный метод

Возможен и другой подход к построению сложных программ: первоначально оставляется множество подпрограмм, которые могут понадобиться при решении задачи, а затем пишется основная программа, содержащая обращения к ним. Подпрограммы могут быть объединены в *библиотеку подпрограмм* и сохранены в долговременной памяти компьютера. Такую библиотеку можно постепенно пополнять новыми подпрограммами.

Например, если для управления графическим исполнителем создать библиотеку процедур рисования всех букв и цифр, то программа получения любого текста будет состоять из команд обращения к библиотечным процедурам.

Описанный метод называется ***сборочным программированием***. Часто в литературе по программированию используется такая терминология: метод последовательной детализации называют ***программированием сверху вниз***, а сборочный метод — ***программированием снизу вверх***.

Коротко о главном:

- Для упрощения программирования сложных задач используются вспомогательные алгоритмы.
- Вспомогательный алгоритм — это алгоритм решения некоторой подзадачи исходной (основной) задачи.
- Вспомогательный алгоритм, записанный на языке программирования, называется процедурой.
- Вспомогательный алгоритм должен быть описан. После этого в основном алгоритме можно использовать команду обращения к этому вспомогательному алгоритму.
- Метод программирования, при котором сначала записывается основной алгоритм, а затем описываются использованные в нем вспомогательные алгоритмы, называется методом последовательной детализации, или программированием сверху вниз. Обратный порядок программирования называется программированием снизу вверх.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §5, вопросы №1-6 к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 5
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №2 Работа с учебным исполнителем алгоритмов: использование вспомогательных алгоритмов.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: обращение к вспомогательному алгоритму (процедуре), описание вспомогательного алгоритма (процедуры)
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практической работы (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практической работы

Ход работы

1. Запускаем программу **Кумир**
2. Выполняем команду меню **Программа – Новая программа**
3. Включаем окно исполнителя **Чертежник**
4. В области написания кода программы для исполнителя набираем:

```
использовать Чертежник
алг Число 2020
нач
. сместиться в точку(-6,5)
. ДВОЙКА
. сместиться на вектор(1,0)
. НОЛЬ
. сместиться на вектор(3,3)
. ДВОЙКА
. сместиться на вектор(1,0)
. НОЛЬ
. сместиться в точку(0,0)
кон

алг ДВОЙКА
нач
. опустить перо
. сместиться на вектор(0,1)
. сместиться на вектор(2,0)
. сместиться на вектор(0,-2)
. сместиться на вектор(-2,-2)
. сместиться на вектор(2,0)
. поднять перо
кон
```

алг НОЛЬ
нач
. опустить перо
. сместиться на вектор(0,4)
. сместиться на вектор(2,0)
. сместиться на вектор(0,-4)
. сместиться на вектор(-2,0)
. поднять перо
кон

5. Запускаем программу на выполнение и проверяем правильность ее работы
6. Дополнительное задание: составить подпрограмму написания слова **ГОД**
6. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §5, задание №7 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 6
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Управление с обратной связью. Язык блок-схем. Использование циклов с предусловием.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: обращение к вспомогательному алгоритму (процедуре), описание вспомогательного алгоритма (процедуры)
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Команда цикла

Обсудим решение следующей задачи.

Задача 3. Исходное положение: ГРИС — у левого края поля, направление — на восток. Требуется нарисовать горизонтальную линию через весь экран.

Задачу можно решить, написав 15 раз команду шаг (если поперек поля рисунка 15 шагов). Но есть и более короткий вариант программы. ***Вот он***:

пока впереди не край **повторять**
нц
шаг
кц

Здесь использована команда, которая называется циклом. Формат команды цикла следующий:

пока <условие> **повторять**
нц
<тело цикла>
кц

Служебное слово **нц** обозначает начало цикла, **кц** — конец цикла. Это первая команда из СКИ, которая реализует обратную связь между графическим исполнителем и управляющим им компьютером. Она заключается в том, что проверяется, не вышел ли ГРИС на край поля, не грозит ли ему следующий шаг или прыжок в этом направлении аварией. Проверяемые условия звучат так: «впереди край?» или «впереди не край?». На что машина получает от исполнителя ответ «да» или «нет».

В приведенном примере проверяется условие «впереди не край?». Если «да», то делается шаг (т. е. выполняется <тело цикла>). Затем происходит возврат на проверку условия, и всё повторяется. Если проверка условия дает отрицательный результат (т. е. впереди край), то выполнение цикла завершается и исполняется следующая после цикла команда программы.

При программировании цикла важно думать о том, чтобы цикл был конечным. Цикл, записанный выше, конечный. Двигаясь в одном направлении, исполнитель обязательно достигнет края, и на этом выполнение цикла закончится.

Ситуация, при которой выполнение цикла никогда не заканчивается, называется **зацикливанием**. Пусть ГРИС находится в середине поля. ***Исполнение следующего цикла***:

пока впереди не край **повторять**
нц
шаг

поворот
кц

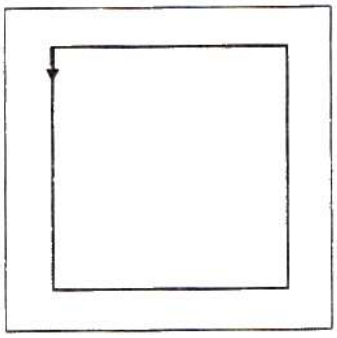
никогда не закончится. ГРИС будет бесконечно рисовать квадратик, так как проверка условия «впереди не край?» всегда будет давать положительный ответ.

Цикл в процедуре

Задача 4. Теперь составим программу, по которой графический исполнитель нарисует прямоугольную рамку по краю поля (рис. 1.6).

Исходное положение: ГРИС находится в левом верхнем углу, смотрит на юг.

Рамка состоит из четырех линий, поэтому разумно воспользоваться процедурой, проводящей линию от края до края поля. Опять будем действовать методом последовательной детализации. Напишем сначала основную программу.



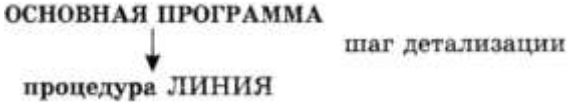
```
программа Рамка
нач
    сделай ЛИНИЯ
    поворот
    сделай ЛИНИЯ
    поворот
    сделай ЛИНИЯ
    поворот
    сделай ЛИНИЯ
кон
```

Рис. 1.6. Результат выполнения программы «Рамка». Стрелкой указано начальное состояние

Программа проведения линии нами уже рассматривалась. Осталось оформить ее в виде процедуры.

процедура ЛИНИЯ
нач
пока впереди не край **повторять**
нц
шаг
кц
кон

При составлении этой программы использовалась одношаговая детализация в такой последовательности:



Блок-схемы алгоритмов

Начиная с 50-х годов прошлого века, т. е. еще с эпохи ЭВМ первого поколения, программисты стали использовать графические схемы, изображающие алгоритмы, которые получили название блок- схем.

Блок-схема состоит из фигур (блоков), обозначающих действия исполнителя, и стрелок, соединяющих эти блоки и указывающих на последовательность их выполнения. Внутри каждого блока записывается выполняемое действие. Форма блока подсказывает характер действия, которое он обозначает. Для придания наглядности и единообразия схемам алгоритмов все графические элементы стандартизированы.

Посмотрите на рис. 1.7, где показана блок-схема алгоритма рисования рамки. Она состоит из двух частей: блок-схемы основного алгоритма и блок-схемы вспомогательного алгоритма ЛИНИЯ.

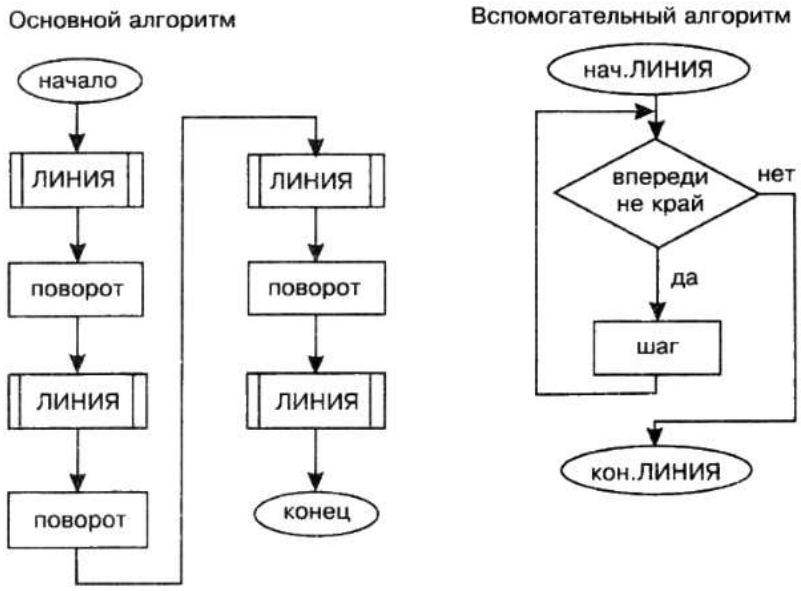


Рис. 1.7. Блок-схема алгоритма «Рамка»

Из этих схем понятно назначение блоков различной формы (рис. 1.8),

Рис. 1.8. Элементы блок-схем и структура «цикл»

Цикл с предусловием

Команда цикла изображается не отдельным блоком, а целой структурой, показанной на рис. 1.8. Такую структуру называют **циклом с предусловием** (так как условие предшествует телу цикла). Есть и другой вариант названия: цикл-пока (пока условие истинно, повторяется выполнение тела цикла).

При решении следующей задачи снова будем использовать метод последовательной детализации.

Задача 5. Требуется расчертить поле горизонтальными линиями (рис. 1.9). Исходное состояние исполнителя: верхний левый угол, направление — на юг.

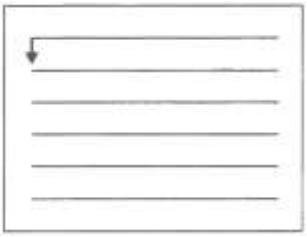


Рис. 1.9. Результат выполнения программы «Разлиновка». Стрелкой указано начальное состояние

В программе для решения этой задачи используется та же процедура **ЛИНИЯ**. Другая процедура — **ВОЗВРАТ** — возвращает ГРИС к левому краю поля для рисования следующей линии.

```
программа Разлиновка
нач
   пока впереди не край
   повторять
   нц
    поворот
    сделай ЛИНИЯ
    сделай ВОЗВРАТ
    прыжок
   кц
   поворот
   сделай ЛИНИЯ
кон
```

```
процедура ВОЗВРАТ
нач
   поворот
   поворот
   пока впереди не край
   повторять
   нц
     прыжок
   кц
   поворот
кон
```

Блок-схемы основного и вспомогательного алгоритмов представлены на рис. 1.10.

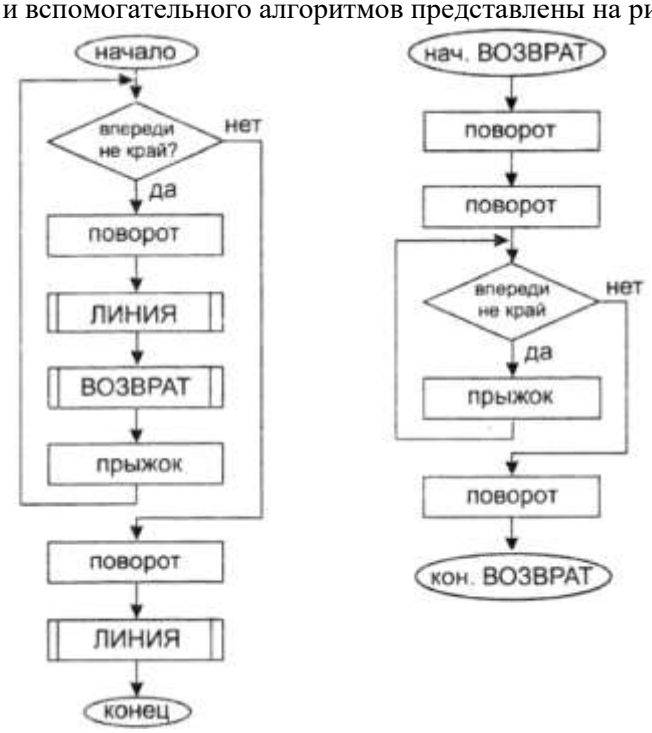


Рис. 1.10. Блок-схема алгоритма «Разлиновка»

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §6, вопросы №1-6 к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 7
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №3 Работа с учебным исполнителем алгоритмов: работа с циклами.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: работа с циклами
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **Кумир**
2. Открываем практический курс **practicum.kurs**
3. Выполняем задания №1 и №2 из подраздела **Циклы n раз**
5. Дополнительное задание: выполняем задание №3 из подраздела **Циклы n раз**
6. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §6 задания №7,8 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 8
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Ветвления. Использование двухшаговой детализации.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: ветвления, использование двухшаговой детализации.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Команда ветвления

Познакомимся еще с одной командой ГРИС. Она называется **командой ветвления**. Формат команды ветвления такой:

если <условие>
то <серия 1>
иначе <серия 2>
кв

Служебное слово **кв** обозначает конец ветвления. По-прежнему ГРИС может проверять только два условия: «впереди край?» или «впереди не край?». <серия> — это одна или несколько следующих друг за другом команд. Если <условие> справедливо, то выполняется <серия 1>, в противном случае — <серия 2>. Такое ветвление называется полным. Пример показан на рис. 1.11.

Рис. 1.11. Блок-схема полного ветвления

Неполная форма команды ветвления

В некоторых случаях используется **неполная форма** команды ветвления (рис. 1.12). Например:
если впереди край
то поворот
КВ

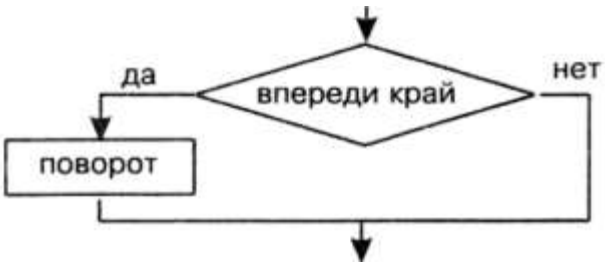


Рис. 1.12. Блок-схема неполного ветвления

Неполная форма команды ветвления:
если <условие>
то <серия>
КВ

Здесь <серия> выполняется, если <условие> справедливо, в противном случае происходит переход к следующей после ветвления команде алгоритма.

Составим последнюю, сравнительно сложную программу для ГРИС. На этом примере вы увидите, что применение метода последовательной детализации облегчает решение некоторых «головоломных» задач.

Пример задачи с двухшаговой детализацией

Задача 6. Построить орнамент, состоящий из квадратов, расположенных по краям поля. Исходное положение ГРИС — в верхнем левом углу, направление — на юг (рис. 1.13).

Процедуру, рисующую цепочку квадратов в одном направлении, назовем РЯД. Процедуру, рисующую один квадрат, назовем КВАДРАТ. Сначала напишем основную программу:

```
программа Орнамент
нач
   сделай РЯД
   поворот
   сделай РЯД
   поворот
   сделай РЯД
   поворот
   сделай РЯД
кон
```

Рис. 1.13. Результат выполнения программы «Орнамент». Стрелкой указано начальное состояние

Теперь напишем процедуры РЯД и КВАДРАТ:

```
процедура РЯД
нач
   прыжок
   прыжок
   пока впереди не край повторять
   нц
     сделай КВАДРАТ
     если впереди не край
           то прыжок
     кв
   кц
кон
```

```
процедура КВАДРАТ
нач
   шаг
   поворот
   шаг
   поворот
   шаг
   поворот
   шаг
   поворот
   прыжок
кон
```

В процедуре РЯД в теле цикла содержится неполное ветвление. Структуру такого алгоритма можно назвать так: цикл с вложенным ветвлением.

На рисунке 1.14 приведена блок-схема процедуры РЯД.

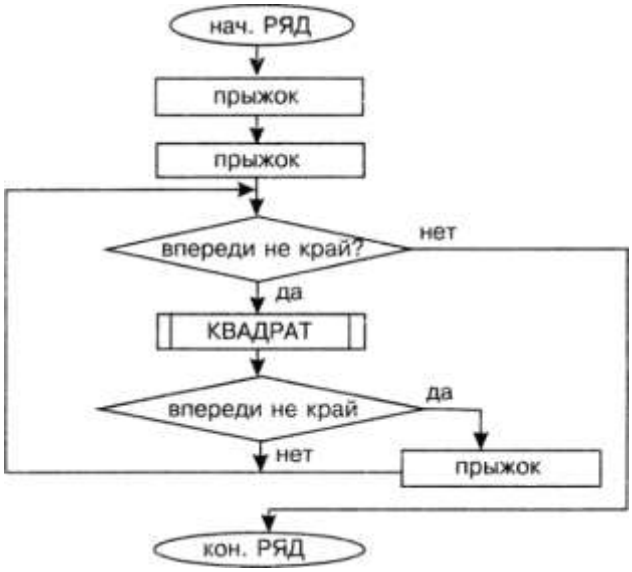


Рис. 1.14. Блок-схема процедуры РЯД

Составление этой программы потребовало двух шагов детализации алгоритма, которые выполнялись в такой последовательности:

ОСНОВНАЯ ПРОГРАММА

1-й шаг детализации

процедура РЯД

2-й шаг детализации

процедура КВАДРАТ

Теперь вам известны все команды управления графическим исполнителем. Их можно разделить на три группы: простые команды; команда обращения к процедуре; структурные команды. К третьей группе относятся команды цикла и ветвления.

СКИ графического исполнителя

Простые команды	Структурные команды
шаг	**пока** <условие> **повторять**
поворот	**нц**
прыжок	<тело цикла>
	кц
Обращение к процедуре	
	если <условие>
сделай <имя процедуры>	**то** <серия 1>
	иначе <серия 2>
	кв

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §7, вопросы №1-4 к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 9
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Использование метода последовательной детализации для построения алгоритма. Использование ветвлений.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: ветвления, использование двухшаговой детализации.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Команда ветвления

Познакомимся еще с одной командой ГРИС. Она называется **командой ветвления**. Формат команды ветвления такой:

если <условие>
то <серия 1>
иначе <серия 2>
кв

Служебное слово **кв** обозначает конец ветвления. По-прежнему ГРИС может проверять только два условия: «впереди край?» или «впереди не край?». <серия> — это одна или несколько следующих друг за другом команд. Если <условие> справедливо, то выполняется <серия 1>, в противном случае — <серия 2>. Такое ветвление называется полным. Пример показан на рис. 1.11.

Рис. 1.11. Блок-схема полного ветвления

Неполная форма команды ветвления

В некоторых случаях используется **неполная форма** команды ветвления (рис. 1.12). Например:
если впереди край
то поворот
КВ

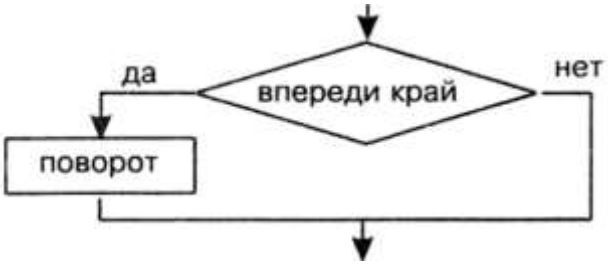


Рис. 1.12. Блок-схема неполного ветвления

Неполная форма команды ветвления:
если <условие>
то <серия>
КВ

Здесь <серия> выполняется, если <условие> справедливо, в противном случае происходит переход к следующей после ветвления команде алгоритма.

Составим последнюю, сравнительно сложную программу для ГРИС. На этом примере вы увидите, что применение метода последовательной детализации облегчает решение некоторых «головоломных» задач.

Пример задачи с двухшаговой детализацией

Задача 6. Построить орнамент, состоящий из квадратов, расположенных по краям поля. Исходное положение ГРИС — в верхнем левом углу, направление — на юг (рис. 1.13).

Процедуру, рисующую цепочку квадратов в одном направлении, назовем РЯД. Процедуру, рисующую один квадрат, назовем КВАДРАТ. Сначала напишем основную программу:

```
программа Орнамент
нач
   сделай РЯД
   поворот
   сделай РЯД
   поворот
   сделай РЯД
   поворот
   сделай РЯД
кон
```

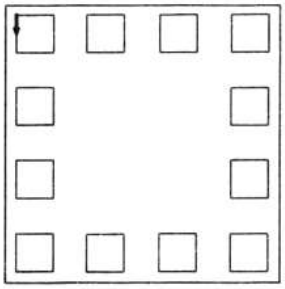


Рис. 1.13. Результат выполнения программы «Орнамент». Стрелкой указано начальное состояние

Теперь напишем процедуры РЯД и КВАДРАТ:

```
процедура РЯД
нач
   прыжок
   прыжок
   пока впереди не край повторять
   нц
     сделай КВАДРАТ
     если впереди не край
          то прыжок
     кв
   кц
кон
```

```
процедура КВАДРАТ
нач
   шаг
   поворот
   шаг
   поворот
   шаг
   поворот
   шаг
   поворот
   прыжок
кон
```

В процедуре РЯД в теле цикла содержится неполное ветвление. Структуру такого алгоритма можно назвать так: цикл с вложенным ветвлением.

На рисунке 1.14 приведена блок-схема процедуры РЯД.

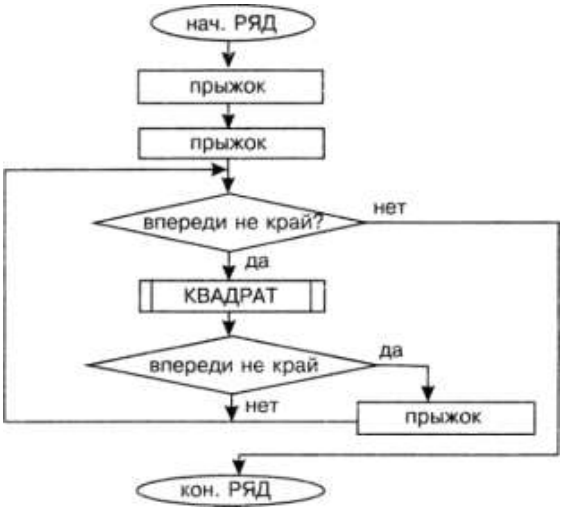


Рис. 1.14. Блок-схема процедуры РЯД

Составление этой программы потребовало двух шагов детализации алгоритма, которые выполнялись в такой последовательности:

ОСНОВНАЯ ПРОГРАММА

1-й шаг детализации

процедура РЯД

2-й шаг детализации

процедура КВАДРАТ

Теперь вам известны все команды управления графическим исполнителем. Их можно разделить на три группы: простые команды; команда обращения к процедуре; структурные команды. К третьей группе относятся команды цикла и ветвления.

СКИ графического исполнителя

Простые команды

шаг
поворот
прыжок

Обращение к процедуре

сделай <имя процедуры>

Структурные команды

пока <условие> **повторять**
нц
<тело цикла>
кц

если <условие>
то <серия 1>
иначе <серия 2>
кв

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §7, вопросы №1-4 к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 10
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №4 Работа с учебным исполнителем алгоритмов: использование ветвлений.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: ветвления, использование двухшаговой детализации.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **Кумир**
2. Открываем практический курс **practicum.kurs**
3. Выполняем задания №1, №2 и №3 из подраздела **Ветвления**
5. Дополнительное задание: выполняем задание №4 из подраздела **Ветвления**
6. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §7, задание №5 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!
Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 11
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №5 Зачётное задание по алгоритмизации.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: ветвления, использование двухшаговой детализации.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **Кумир**
2. Открываем практический курс **practicum.kurs**
3. Выполняем задания №1 №2 по вариантам из подраздела **Сложные условия**
5. Дополнительное задание: выполняем задание №3 из подраздела **Сложные условия**
6. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§1-*7*, подготовка к КР №1.
Комментарии учителя.

Урок окончен, до свидания!
Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 12
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Контрольная работа №1 Итоговое тестирование по теме «Управление и алгоритмы».
Тип урока: Итоговый контроль и учет знаний и навыков.
Цели урока:
Образовательные
Изучение вопросов: ветвления, использование двухшаговой детализации.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Итоговое тестирование по теме «Управление и алгоритмы» (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Зачётное задание по алгоритмизации

В браузере Интернет перейти по адресу:
https://иванов-ам.рф/informatika_09_sim/ur_21/test_upravlenie_i_algoritmyi.html

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§1-7, повторить.
Комментарии учителя.

Урок окончен, до свидания!
Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 13
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Понятие о программировании. Алгоритмы работы с величинами. Линейные вычислительные алгоритмы.
Тип урока: Урок-лекция.
Цели урока:
Образовательные
Изучение вопросов: понятие о программировании, алгоритмы работы с величинами, линейные вычислительные алгоритмы.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Кто такие программисты

Теперь вам предстоит ближе познакомиться еще с одним разделом информатики, который называется «Программирование».

Назначение программирования — разработка программ управления компьютером с целью решения различных информационных задач.

Специалисты, профессионально занимающиеся программированием, называются **программистами**. В первые годы существования ЭВМ для использования компьютера в любой области нужно было уметь программировать. В 1970-1980-х годах начинает развиваться прикладное программное обеспечение. Бурное распространение прикладного ПО произошло с появлением персональных компьютеров. Стало совсем не обязательным уметь программировать для того, чтобы воспользоваться компьютером. Люди, работающие на компьютерах, разделились на **пользователей и программистов**. В настоящее время пользователей гораздо больше, чем программистов.

Может возникнуть впечатление, что программисты теперь уже и не нужны! Но кто же тогда будет создавать все операционные системы, редакторы, графические пакеты, компьютерные игры и многое другое? Программисты, безусловно, нужны, причем задачи, которые им приходится решать, со временем становятся все сложнее.

Программирование принято разделять на системное и прикладное. **Системные программисты** занимаются разработкой системного программного обеспечения: операционных систем, утилит и пр., а также систем программирования. **Прикладные программисты** создают прикладные программы: редакторы, табличные процессоры, игры, обучающие программы и др. Спрос на высококвалифицированных программистов, как системных, так и прикладных, очень большой.

В данной главе вы познакомитесь с простейшими правилами и приемами программирования, заглянете в эту актуальную и престижную профессиональную область.

Что такое язык программирования

Для составления программ существуют разнообразные **языки программирования**.

Язык программирования — это фиксированная система обозначений для описания алгоритмов и структур данных.

За годы существования ЭВМ было создано много языков программирования. Наиболее известные среди них: Фортран, Паскаль, Бейсик, С (Си) и др.

Распространенными языками программирования сегодня являются C++, Java, Pascal, Basic, Python.

Что такое система программирования

Для создания и исполнения на компьютере программы, написанной на языке программирования, используются **системы программирования**.

Система программирования — это программное обеспе чение компьютера, предназначенное для разработки, от ладки и исполнения программ, записанных на определен ном языке программирования.

Существуют системы программирования на Паскале, Бейсике и других языках.

В данной главе речь будет идти о средствах и способах универсального программирования — не ориентированного на какую-то узкую прикладную область. Примером узкоспециализированного программирования является Web-программирование, ориентированное на создание Web-сайтов. Для этих целей, например, используется язык JavaScript. Языки Паскаль, Бейсик, Си относятся к числу **универсальных языков программирования**.

Разработка любой программы начинается с построения алгоритма решения задачи. Ниже мы обсудим особенности алгоритмов решения задач обработки информации на компьютере.

Компьютер как исполнитель алгоритмов

Вам уже известно, что всякий алгоритм составляется для конкретного исполнителя. Теперь в качестве исполнителя мы будем рассматривать компьютер, оснащенный системой программирования на определенном языке.

Компьютер-исполнитель работает с определенными данными по определенной программе. **Данные — это множество величин**.

Величины: константы и переменные

Компьютер работает с информацией, хранящейся в его памяти. Отдельный информационный объект (число, символ, строка, таблица и пр.) называется величиной.

Всякая обрабатываемая программой величина занимает свое место (поле) в памяти компьютера.

Значение величины — это информация, хранимая в этом поле памяти.

Существуют три основных типа величин, с которыми работает компьютер: **числовой, символьный и логический**. Изучая базы данных и электронные таблицы, вы уже встречались с этими типами. В данной главе мы будем строить алгоритмы, работающие с числовыми величинами.

Числовые величины в программировании, так же как и математические величины, делятся на переменные и константы (постоянные). Например, в формуле $(a^2 - 2ab + b^2)$ a, b — переменные, 2 — константа.

Константы записываются в алгоритмах своими десятичными значениями, например: 23, 3.5, 34. Значение константы хранится в выделенной под нее ячейке памяти и остается неизменным в течение работы программы.

Переменные в программировании, как и в математике, обозначаются символическими именами. Эти имена называют идентификаторами (от глагола «идентифицировать», что значит «обозначать», «символизировать»). Идентификатор может быть одной буквой, множеством букв, сочетанием букв и цифр и т. д. Примеры идентификаторов: A, X, B3, prim, r25 и т. п.

Система команд

Вам известно, что всякий алгоритм строится исходя из системы команд исполнителя, для которого он предназначен. Любой алгоритм работы с величинами может быть составлен из следующих команд:

- присваивание;
- ввод;
- вывод;
- обращение к вспомогательному алгоритму;
- цикл;
- ветвление.

Эти команды существуют во всех языках, поддерживающих структурное программирование: в Паскале, Си и др.

Команда присваивания

Команда присваивания — одна из основных команд в алгоритмах работы с величинами. Записывать ее мы будем так:

<переменная>:=<выражение>

Значок «:=» читается «присвоить». Например:

Z:=X + Y

Компьютер сначала вычисляет выражение, затем результат присваивает переменной, стоящей слева от знака «:=».

Если до выполнения этой команды содержимое ячеек, соответствующих переменным **X, Y, Z,** было таким:

то после выполнения команды оно станет следующим:

Прочерк в ячейке **Z** обозначает, что начальное число в ней может быть любым. Оно не имеет значения для результата данной команды.

Если слева от знака присваивания стоит числовая переменная, а справа — выражение, определяющее порядок вычисления числовой величины, то такую команду называют арифметической командой присваивания, а выражение — арифметическим выражением.

В частном случае арифметическое выражение в правой части оператора присваивания может быть представлено одной переменной или одной константой.

Например:

X:= 5

Y:= X

Команда ввода

Значения переменных, являющихся исходными данными решаемой задачи, как правило, задаются вводом.

Команда ввода в описаниях алгоритмов выглядит так:

ввод <список переменных>.

Например:

ввод **А, Б, С**

Пользователю удобно, если ввод данных организован в режиме диалога, когда по команде ввода компьютер прерывает выполнение программы и ждет действий пользователя. Пользователь должен набрать на клавиатуре вводимые значения переменных и нажать клавишу <ВВОД>. Введенные значения присвоятся соответствующим переменным из списка ввода, и выполнение программы продолжится.

Вот схема выполнения приведенной выше команды.

1. Память до выполнения команды:

2. Процессор компьютера получил команду **ввод А, Б, С**, прервал свою работу и ждет действий пользователя.

3. Пользователь набирает на клавиатуре:

1 3 5

и нажимает клавишу <ВВОД> (<Enter>).

4. Память после выполнения команды:

5. Процессор переходит к выполнению следующей команды программы.

При выполнении пункта 3 вводимые числа должны быть отделены друг от друга какими-нибудь разделителями. Обычно это пробелы. Из сказанного выше можно сделать вывод:

Переменные величины получают конкретные значения в результате выполнения команды присваивания или команды ввода.

Если переменной величине не присвоено никакого значения (или не введено), то она является неопределенной. Иначе говоря, ничего нельзя сказать о том, какое значение имеет эта переменная.

Команда вывода

Результаты решения задачи сообщаются компьютером пользователю путем выполнения команды вывода.

Команда вывода в алгоритмах записывается так:

вывод <список вывода>

Например:

вывод **X1, X2**

По этой команде значения переменных **X1 и X2** будут вынесены на устройство вывода (чаще всего это экран).

О других командах, применяемых в алгоритмах работы с величинами, вы узнаете позже.

Присваивание. Свойства присваивания

Поскольку присваивание является важнейшей операцией в алгоритмах, работающих с величинами, поговорим о ней более подробно.

Переменная величина получает значение в результате присваивания.

Присваивание производится компьютером при выполнении одной из двух команд из представленной выше системы команд: команды присваивания или команды ввода.

Команда	a	b
a:=1	1	—
b:=2 * a	1	2
a:=b	2	2
b:=a + b	2	4

Рассмотрим последовательность выполнения четырех команд присваивания, в которых участвуют две переменные: **a и b**. В приведенной ниже таблице против каждой команды указываются значения переменных, которые устанавливаются после ее выполнения. Такая таблица называется **трассировочной таблицей**, а процесс ее заполнения называется **трассировкой алгоритма**.

Прочерк в таблице означает неопределенное значение переменной. Конечные значения, которые получают переменные **a и b**, соответственно равны **2 и 4**.

Этот пример иллюстрирует три основных свойства присваивания. Вот эти свойства:

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

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

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

Обмен значениями двух переменных

Рассмотрим еще один очень полезный алгоритм, с которым при программировании часто приходится встречаться. Даны две переменные величины: **X и Y**. Требуется произвести между ними обмен значениями. Например, если первоначально было: **X = 1; Y = 2**, то после обмена должно стать: **X = 2, Y = 1**.

Хорошим аналогом для решения такой задачи является следующая: даны два стакана, в первом — молоко, во втором — вода; требуется произвести обмен их содержимым. Всякому ясно, что в этом случае нужен дополнительный, третий, пустой стакан. Последовательность действий будет следующей:

1) перелить из 1-го стакана в 3-й;

2) перелить из 2-го стакана в 1-й;

3) перелить из 3-го стакана во 2-й.

Цель достигнута!

Команда	X	Y	Z
ввод X, Y	1	2	–
Z:=X	1	2	1
X:=Y	2	2	1
Y:=Z	2	1	1
вывод X, Y	**2**	**1**	1

По аналогии для обмена значениями двух переменных нужна третья дополнительная переменная. Назовем ее Z. Тогда задача решается последовательным выполнением трех операторов присваивания (пусть начальные значения 1 и 2 для переменных X и Y задаются вводом):

Действительно, в итоге переменные **X и Y** поменялись значениями. На экран будут выведены значения **X и Y: 2,1**. В трассировочной таблице выводимые значения выделены жирным шрифтом.

Аналогия со стаканами не совсем точна в том смысле, что при переливании из одного стакана в другой первый становится пустым. В результате же присваивания (**X:=Y**) переменная, стоящая справа (**Y**), сохраняет свое значение.

Описание линейного вычислительного алгоритма

Алгоритмы, результатами выполнения которых являются числовые величины, будем называть вычислительными алгоритмами. Рассмотрим пример решения следующей математической задачи:

даны две простые дроби; получить дробь, являющуюся результатом деления одной на другую.

В школьном учебнике математики правила деления обыкновенных дробей описаны так:

1. Числитель первой дроби умножить на знаменатель второй.

2. Знаменатель первой дроби умножить на числитель второй.

3. Записать дробь, числителем которой является результат выполнения пункта 1, а знаменателем — результат выполнения пункта 2.

В алгебраической форме это выглядит следующим образом:

$$\frac{a}{b} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c} = \frac{m}{n}.$$

Теперь построим алгоритм деления дробей для компьютера. В этом алгоритме сохраним те же обозначения для переменных, которые использованы в записанной выше формуле. Исходными данными являются целочисленные переменные **a, b, c, d**. Результатом — также целые величины **m** и **n**.

Ниже алгоритм представлен в двух формах: в виде блок-схемы и на Алгоритмическом языке (АЯ).

Раньше прямоугольник в схемах алгоритмов управления мы называли блоком простой команды. Для вычислительных алгоритмов такой простой командой является команда присваивания. Прямоугольник будем называть **блоком присваивания**, или **вычислительным блоком**. В форме параллелограмма рисуется **блок ввода/вывода**. Полученный алгоритм имеет линейную структуру (рис. 2.1).

```
алг  Деление дробей
цел  a, b, c, d, m, n
нач
   ввод a, b, c, d
   m:=a×d
   n:=b×c
   вывод m, n
кон
```

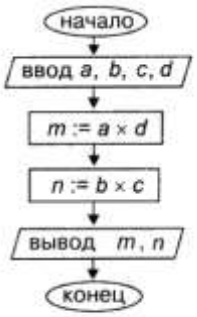


Рис. 2.1. Блок-схема алгоритма деления дробей

В алгоритме на АЯ строка, стоящая после заголовка алгоритма, называется **описанием переменных**. Служебное слово цел означает целый тип. Величины этого типа могут иметь только целочисленные значения.

Описание переменных имеет вид:

<тип переменных> <список переменных>

Список переменных включает все переменные величины данного типа, обрабатываемые в алгоритме.

В блок-схемах типы переменных не указываются, но подразумеваются. Запись алгоритма на АЯ ближе по форме к языкам программирования, чем блок-схемы.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§8-10, вопросы к параграфам устно.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 14
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Возникновение и назначение языка Паскаль. Структура программы на языке Паскаль. Операторы ввода, вывода, присваивания.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: Возникновение и назначение языка Паскаль. Структура программы на языке Паскаль. Операторы ввода, вывода, присваивания.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Возникновение и назначение Паскаля

После того как построен алгоритм решения задачи, составляется программа на определенном языке программирования.

Среди современных языков программирования одним из самых популярных является язык Паскаль. Этот язык разработан в 1971 году и назван в честь Влеза Паскаля — французского ученого, изобретателя механической вычислительной машины. Автор языка Паскаль — швейцарский профессор Никлаус Вирт.

Паскаль — это универсальный язык программирования, позволяющий решать самые разнообразные задачи обработки информации.

Команду алгоритма, записанную на языке программирования, принято называть оператором.

Программа на Паскале близка по своему виду к описанию алгоритма на АЯ. Сравните алгоритм решения уже знакомой вам задачи деления простых дробей с соответствующей программой на Паскале:

```
алг Деление дробей
цел a, b, c, d, m, n
нач
   ввод a, b, c, d
   m:= a×d
   n:= b×c
   вывод m, n
кон
```

```
Program Division;
 var a, b, c, d, m, n: integer;
 begin
    readln(a, b, c, d);   {Ввод}
     m:= a*d;   {Числитель}
     n:= b*c;   {Знаменатель}
     write(m, n)   {Вывод}
 end.
```

Структура программы на Паскале

Даже не заглядывая в учебник по Паскалю, в этой программе можно все понять (особенно помогает знание английского языка).

Заголовок программы начинается со слова Program (программа), за которым следует произвольное имя, придуманное программистом:

Program <имя программы>;

Раздел описания переменных начинается со слова **Var** (variables — переменные), за которым идет список имен переменных через запятую. Тип указывается после двоеточия. В стандарте языка Паскаль существуют два типа числовых величин: **вещественный и целый**. Слово **integer** обозначает целый тип (является идентификатором целого типа). Вещественный тип обозначается словом **real**. Например, раздел описания переменных может быть таким:

var a, b: integer; c, d: real;

Идентификаторы переменных составляются из латинских букв и цифр; первым символом обязательно должна быть буква.

Раздел операторов — основная часть программы. Начало и конец раздела операторов программы отмечаются служебными словами **begin** (начало) и **end** (конец). В самом конце программы ставится точка:

begin
<операторы>
end.

Операторы ввода, вывода, присваивания

Ввод исходных данных с клавиатуры происходит по оператору **read** (**read** — читать) или **readln** (**read line** — читать строку):

read(<список переменных>)
или readln(<список переменных>)

При выполнении команды ввода компьютер ожидает действий пользователя. Пользователь набирает на клавиатуре значения переменных в том порядке, в каком переменные указаны в списке, отделяя их друг от друга пробелами. Одновременно с набором данных на клавиатуре они появляются на экране. В конце нажимается клавиша <ВВОД> (<Enter>). Разница в выполнении операторов readln и read состоит в том, что после выполнения ввода по оператору readln экранный курсор перемещается в начало новой строки, а по оператору read этого не происходит.

Вывод результатов происходит по оператору **write** (**write** — писать) или **writeln** (**write line** — писать в строку):

write(<список вывода>)
или writeln(<список вывода>)

Результаты выводятся на экран компьютера в порядке их перечисления в списке. Элементами списка вывода могут быть константы, переменные, выражения.

Разница в выполнении операторов writeln и write состоит в том, что после выполнения вывода по оператору writeln экранный курсор перемещается в начало новой строки, а по оператору write этого не происходит.

Арифметический оператор присваивания на Паскале имеет следующий формат:

<числовая переменная>:=<арифметическое выражение>

Арифметическое выражение может содержать числовые константы и переменные, знаки арифметических операций, круглые скобки. Кроме того, в арифметических выражениях могут присутствовать функции.

Знаки основных арифметических операций записываются так:

+ сложение,
- вычитание,
* умножение,
/ деление.

Правила записи арифметических выражений

Запись арифметических выражений на Паскале похожа на обычную математическую запись. В отличие от математики, где часто пропускается знак умножения (например, пишут 2A), в Паскале этот знак пишется обязательно: 2*A. Например, математическое выражение

$\mathbf{A^2 + B^2 - 12C}$

на Паскале записывается так:

A*A + B*B - 12*C

Это же выражение можно записать иначе:

SQR(A) + SQR(B) - 12*C

Здесь использована функция возведения в квадрат — **SQR**. Аргументы функций всегда пишутся в круглых скобках.

Последовательность выполнения операций определяется по их **приоритетам** (старшинству). К старшим операциям относятся умножение (*) и деление (/). Операции сложения и вычитания — младшие. В первую очередь выполняются старшие операции. Несколько операций одинакового старшинства, записанные подряд, выполняются в порядке их записи слева направо. Приведенное выше арифметическое выражение будет вычисляться в следующем порядке (порядок вычислений указан цифрами сверху):

```
  1   4   2   5     3
A * A + B * B - 12 * C
```

Круглые скобки в арифметических выражениях влияют на порядок выполнения операций. Как и в математике, в первую очередь выполняются операции в скобках. Если имеется несколько пар вложенных скобок, то сначала выполняются операции в самых внутренних скобках. Например:

```
      6     1     3     2     4     5
A + ((C - D) / (2 + K) - 1)*B
```

Пунктуация Паскаля

Необходимо строгое соблюдение правописания (синтаксиса) программы. В частности, ***в Паскале однозначно определено назначение знаков пунктуации***.

Точка с запятой (;) ставится в конце заголовка программы, в конце раздела описания переменных, является разделителем описания переменных в разделе переменных и разделителем операторов. Перед словом **end** точку с запятой можно не ставить.

Запятая (,) является разделителем элементов во всевозможных списках: списке переменных в разделе описания, списках вводимых и выводимых величин.

Текст программы заканчивается **точкой**.

Строгий синтаксис в языке программирования необходим потому, что **компьютер является формальным исполнителем программы**. Если, допустим, разделителем в списке переменных должна быть запятая, то любой другой знак будет восприниматься как ошибка. Если точка с запятой является разделителем операторов, то в качестве оператора компьютер воспринимает всю часть текста программы от одной точки с запятой до другой. Если программист забыл поставить «;» между какими-то двумя операторами, то компьютер будет принимать их за один с неизбежной ошибкой.

В программу на Паскале можно вставлять комментарии. Комментарий — это пояснение к программе, которое записывается в фигурных скобках. В комментариях можно использовать русские буквы. На исполнение программы комментарий никак не влияет.

Заметим, что **в Паскале нет различия между строчными и прописными буквами**. Например, для Паскаля тождественны следующие варианты записи: **begin, Begin, BEGIN, BeGiN**. Использование строчных или прописных букв — дело вкуса программиста.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §11, задание №9 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 15
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №6 Работа с готовыми программами на языке Паскаль: отладка, выполнение, тестирование.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Возникновение и назначение языка Паскаль. Структура программы на языке Паскаль. Операторы ввода, вывода, присваивания.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **PascalABC.NET**
2. Открываем готовую программу по адресу: C:\PABCWork.NET\Samples\!Tutorial\01_First\ AssignExt.pas
3. Изучить назначение программы
4. Выполнить (запустить) программу и просмотреть результат ее работы
5. Внести изменения в программу (заведомо не корректные), выполнить и в режиме отладки исправить ошибки
6. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §11, вопросы №1-8 к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!
Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 16
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №7 Программирование на Паскале линейных алгоритмов.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Программирование на Паскале линейных алгоритмов.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **PascalABC.NET**
2. Создаем новый документ и выполняем следующие задания:

а) Разработать схему алгоритма и программу, которая **присваивает целой** переменной **A** значение **10** и **выводит** это значение на экран. Отладить созданную программу.

б) Разработать схему алгоритма и программу, которая запрашивает **ввод целого** числа в переменную **B** и **выводит** это число на экран. **Отладить** алгоритм и проверить **правильность** его работы на числах **1, -5, 256, 10455**.

в) Разработать схему алгоритма и программу для ввода **четырёх целых чисел** и вычисления их **среднего арифметического**. **Протестировать** алгоритм на различных исходных данных (включая вещественные числа) и **доказать** правильность его работы.

3. Дополнительное задание: Вводятся величины **X,Y целого** типа. Разработать схему алгоритма для **обмена** значений величин. Необходимо использовать **вспомогательную** величину **T**. **Протестировать** алгоритм для **X=5 и Y=-11**.
4. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §11, задание №10 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!
Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 17
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Оператор ветвления. Логические операции на Паскале. Программирование ветвлений на Паскале. Программирование диалога с компьютером.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: Оператор ветвления. Логические операции на Паскале. Программирование ветвлений на Паскале. Программирование диалога с компьютером.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Представление ветвлений на АЯ. Трассировка ветвящихся алгоритмов

Рассмотрим несколько задач, решение которых на компьютере получается с помощью ветвящихся алгоритмов.

Первая задача: **даны два числа; выбрать большее из них**.

Пусть исходными данными являются переменные **A** и **B**. Их значения будут задаваться вводом. Значение большего из них должно быть присвоено переменной **C** и выведено на экран компьютера. Например, если **A = 5, B = 8**, то должно получиться: **C = 8**.

Блок-схема алгоритма решения этой задачи изображена на рис. 2.2.

Нетрудно понять смысл этого алгоритма. Если значение переменной **A** больше, чем **B**, то переменной **C** присвоится значение **A**. В противном случае, когда **A ≤ B**, переменной **C** присвоится значение **B**.

Условием, по которому разветвляется алгоритм, является отношение неравенства **A > B**. Изучая базы данных и электронные таблицы, вы узнали, что такое отношение является логическим выражением. Если оно справедливо, то результатом будет логическая величина «истина» и выполнение алгоритма продолжится по ветви «да»; в противном случае логическое выражение примет значение «ложь» и выполнение алгоритма пойдет по ветви «нет».

До выполнения на компьютере правильность алгоритма можно проверить путем заполнения трассировочной таблицы. Вот как будет выглядеть трассировка нашего алгоритма для исходных значений **A = 5, B = 8**.

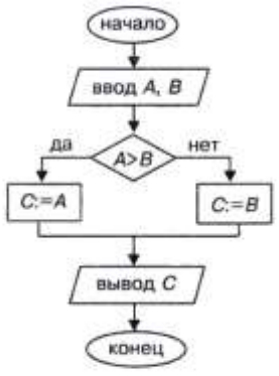


Рис. 2.2. Алгоритм выбора большего из двух чисел (с полным ветвлением)

Шаг	Операция	A	B	C	Проверка условия
1	ввод A, B	5	8	–	
2	A>B	5	8	–	5 > 8, нет (ложь)
3	C:=B	5	8	8	
4	вывод C	5	8	8	

Ветвление является **структурной командой**. Его исполнение происходит в несколько шагов: проверка условия (выполнения логического выражения) и выполнение команд на одной из ветвей «да» или «нет». Поэтому в трассировочной таблице записываются не команды алгоритма, а отдельные операции, выполняемые компьютером на каждом шаге.

В алгоритме на рис. 2.2 используется **полное ветвление**. Эту же самую задачу можно решить, применяя структурную команду **неполного ветвления**. Блок-схема такого алгоритма изображена на рис. 2.3.

Выполните самостоятельно трассировку этого алгоритма для вариантов 1) A = 0,2, B = 0,3; 2) A = 7, Б = 4; 3) A = 5, B = 5. Если вы всё проделаете правильно, то убедитесь, что алгоритм верный.

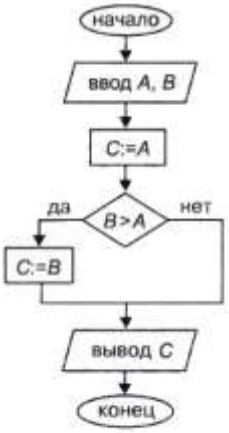


Рис. 2.3. Алгоритм выбора большего из двух значений (с неполным ветвлением)

А теперь запишем рассмотренные алгоритмы на АЯ. Во-первых, нужно решить вопрос о том, как описать переменные в этом алгоритме. Для всех переменных в алгоритме на АЯ необходимо указать их тип.

Переменные **A, B, C** — числовые величины. В этой задаче они могут принимать любые значения. В программировании числовые величины, которые могут иметь любые значения — целые, дробные, называются вещественными. Им ставится в соответствие вещественный тип. На АЯ этот тип указывается служебным словом вещ.

Как выглядит команда ветвления, вы уже знаете. Вот два алгоритма на АЯ, соответствующие блок-схемам на рис. 2.2 и 2.3.

```
алг БИД1
вещ A, B, C
нач
   ввод A, B
   если A>B
        то C:=A
        иначе C:=B
   кв
   вывод C
кон
```

```
алг БИД2
вещ A, B, C
нач
   ввод A, B
   C:=A
   если B>A
        то C:=B
   кв
   вывод C
кон
```

Под сокращенным названием алгоритмов ВИД подразумевается «Большее из двух».

Для программирования характерно то, что одна и та же задача может быть решена с помощью разных алгоритмов. И чем сложнее задача, тем больше можно придумать различных алгоритмов ее решения. Для больших задач (производственных, научных) практически невозможно точное совпадение алгоритмов, составленных разными программистами.

Следующая задача: **упорядочить значения двух переменных X и Y по возрастанию**. Смысл этой задачи следующий: если для исходных значений переменных справедливо отношение **X ≤ Y** (например, **X = 1, Y = 2**), то оставить их без изменения; если же **X > Y** (например, **X = 2, Y = 1**), то выполнить обмен значениями.

Алгоритм обмена значениями двух переменных был рассмотрен в предыдущем параграфе. Вспомним, что для обмена мы использовали третью, вспомогательную переменную.

В алгоритме решения данной задачи используется неполное ветвление. Приведем блок-схему (рис. 2.4) и алгоритм на АЯ.

Здесь роль вспомогательной переменной для обмена выполняет **C**.

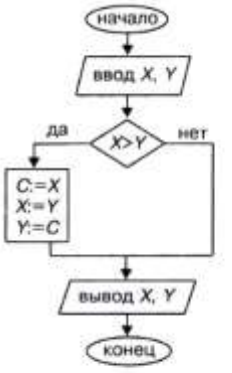


Рис. 2.4. Блок-схема алгоритма упорядочения двух величин

```
алг СОРТИРОВКА
вещ X, Y, C
нач
   ввод X, Y
   если X>Y
        то C:=X
           X:=Y
           Y:=C
   кв
   вывод X, Y
кон
```

Сложные ветвящиеся алгоритмы

Получим алгоритм решения еще одной задачи: ***найти наибольшее значение среди трех величин***: **A, B, C**.

Естественно, возникает следующая идея этого алгоритма: сначала нужно найти большее из значений **A** и **B** и присвоить его какой-то дополнительной переменной, например **D**; затем найти большее среди **D** и **C**. Это значение можно присвоить той же переменной **D.**

Решение задачи сводится к двукратному применению уже знакомого алгоритма нахождения большего из двух значений. Блок-схема алгоритма — на рис. 2.5.

```
алг БИТ1
вещ A, B, C, D
нач
    ввод A, B, C
    если A>B
        то D:=A
        иначе D:=B
    кв
    если C>D
        то D:=C
    кв
    вывод D
кон
```

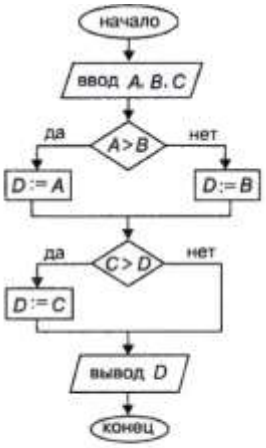


Рис. 2.5. Блок-схема алгоритма «БИТ» с последовательными ветвлениями

Нетрудно догадаться, что «БИТ» обозначает «Большее из трех». В структуре этого алгоритма содержатся два последовательных ветвления: первое — полное, второе — неполное.

Эту же задачу можно решить с помощью алгоритма, имеющего структуру вложенных ветвлений. Его блок-схема приведена в следующем параграфе на рис. 2.6. А вот как выглядят описание этого алгоритма на АЯ и трассировочная таблица при **A = 5, B = 7, C = 2**.

```
алг БИТ2
вещ A, B, C, D
нач
    ввод A, B, C
    если A>B
        то если A>C то D:=A иначе D:=C кв
        иначе если B>C то D:=B иначе D:=C кв
    кв
    вывод D
кон
```

Шаг	Операция	A	B	C	D	Проверка условия
1	ввод A, B, C	5	7	2	–	
2	A>B	5	7	2	–	5 > 7, нет
3	B>C	5	7	2	–	7 > 2, да
4	D:=B	5	7	2	7	
5	вывод D	5	7	2	7	

Оператор ветвления на Паскале

В языке Паскаль имеется оператор ветвления. Другое его название — условный оператор.

Формат полного оператора ветвления следующий:

if <логическое выражение> **then** <оператор1> **else** <оператор2>

Здесь **if** — «если», **then** — «то», **else** — «иначе».

Программирование полного и неполного ветвления

Сравните запись алгоритма БИД 1 из предыдущего параграфа с соответствующей программой.

```
алг БИД1
вещ A, B, C
нач
   ввод A, B
   если A>B
        то C:=A
        иначе C:=B
   кв
   вывод C
кон
```

```
Program BID1;
var A, B, C: real;
begin
  readln(A, B);
  if A>B
  then C:=A
  else C:=B;

  writeln(C)
end.
```

Очень похоже на перевод с русского языка на английский. Обратите внимание на следующее отличие: в программе нет специального служебного слова, обозначающего конец ветвления. Здесь признаком конца оператора ветвления является точка с запятой. (Разумеется, оставлять в программе пустую строку совсем не обязательно. Здесь это сделано только ради наглядности.)

Простой формой логического выражения является операция отношения. Как и в АЯ, в Паскале допускаются все виды отношений (ниже указаны их знаки):

< (меньше);	<= (больше или равно);
> (больше);	= (равно);
<= (меньше или равно);	<> (не равно).

А теперь запрограммируем на Паскале алгоритм БИД2, в котором использовано неполное ветвление.

```
алг БИД2
вещ A, B, C
нач
   ввод A, B
   C:=A
   если B>A
        то C:=B
   кв
   вывод C
кон
```

```
Program BID2;
var  A, B, C : real;
begin
  readln(A, B);
  C:=A;
  if  B>A
  then C:=B;

  write(C)
end.
```

Опять всё очень похоже. Ветвь else в операторе ветвления может отсутствовать.

Программирование вложенных ветвлений

Запишем на Паскале программу определения большего из трех чисел, блок-схема которой показана на рис. 2.6. Структура этого алгоритма — вложенные ветвления. Алгоритм на АЯ (БИТ2) приведен в предыдущем параграфе.

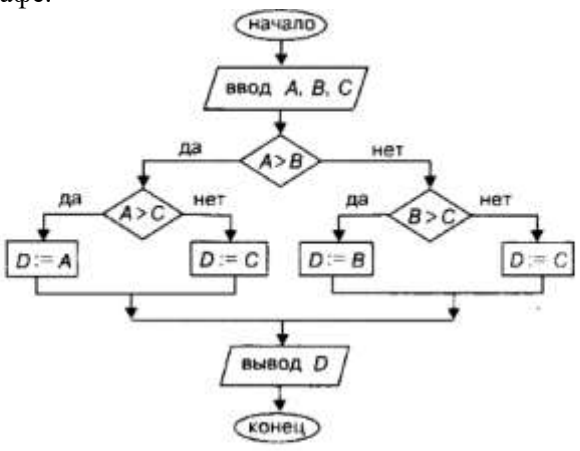


Рис. 2.6. Блок-схема алгоритма «БИТ» с вложенными ветвлениями

```
Program BIT2;
var A, B, C, D: real;
begin
   readln(A, B, C);
   if A>B
      then if A>C then D:=A else D:=B
      else if B>C then D:=B else D:=C;
   writeln(D)
end.
```

Обратите внимание на то, что перед else символ «;» не ставится, так как этот символ является разделителем операторов. Вся ветвящаяся часть структуры алгоритма заканчивается на точке с запятой после оператора **D:=C**.

Составим программу упорядочения значений двух переменных.

```
алг СОРТИРОВКА
вещ X, Y, C
нач
   ввод X, Y
   если X>Y
        то C:=X
           X:=Y
           Y:=C
   кв
   вывод X, Y
кон
```

```
Program SORTING;
var X, Y, C: real;
begin
  readln(X, Y);
  if  X>Y
  then begin C:=X;
         X:=Y;
         Y:=C
       end;
write(X,Y)
end.
```

Этот пример иллюстрирует следующее правило Паскаля: если на какой-то из ветвей оператора ветвления находится несколько последовательных операторов, то их нужно записывать между служебными словами **begin** и **end**. Конструкция такого вида:

begin <последовательность операторов> **end**

называется **составным оператором**. Следовательно, в описанной выше общей форме ветвления <оператор1> и <оператор2> могут быть простыми и составными операторами.

Логические операции

Наконец, составим еще один, третий вариант программы определения наибольшего числа из трех.

```
Program BIT3;
var A, B, C, D: real;
begin
      readln(A,B,C);
      if (A>=B) and (A>=C) then D:=A;
      if (B>=A) and (B>=C) then D:=B;
      if (C>=A) and (C>=B) then D:=C;
      writeln(D)
end.
```

Нетрудно понять смысл этой программы. Здесь использованы три последовательных неполных ветвления. А условия ветвлений представляют собой **сложные логические выражения**, включающие **логическую операцию and (И)**. С логическими операциями вы встречались, работая с базами данных и с электронными таблицами.

Напомним, что операция and называется **логическим умножением или конъюнкцией**. Ее результат — «истина», если значения обоих операндов — «истина». Очевидно, что если **A ≥ B и A ≥ C**, то **A** имеет наибольшее значение и т. д. В Паскале присутствуют все три основные логические операции:

and — И (конъюнкция), or — ИЛИ (дизъюнкция), not — НЕ (отрицание).

Замечание. Мы рассмотрели три варианта решения задачи о поиске максимального числа из трех заданных (БИТ1, БИТ2, БИТ3). Предложенные варианты решения, помимо всего прочего, отличаются еще и эффективностью: эффективность тем выше, чем меньше исполняется операций при выполнении программы. С этой точки зрения, самым неэффективным является алгоритм БИТ3. В любом случае в нем будет выполняться 6 операций отношения. При исполнении алгоритмов БИТ1 и БИТ2 выполняются 2 операции отношения. Понятно, что на практике необходимо использовать наиболее эффективные алгоритмы.

Сложные логические выражения

Обратите внимание на то, что отношения, связываемые логическими операциями, заключаются в скобки. Так надо делать всегда! Например, требуется определить, есть ли среди чисел **A, B, C** хотя бы одно отрицательное. Эту задачу решает следующий оператор ветвления:

if (A<0) **or** (B<0) **or** (C<0) **then** write ('YES ') **else** write ('NO ')

Выражение, истинное для отрицательного числа, может быть записано еще и так:

not(A>=0)

Что такое диалог с компьютером

Если вы исполняли рассмотренные выше программы на компьютере, то почувствовали определенное неудобство при работе с машиной. **Во-первых**, непонятно, когда машина начинает ожидать ввода данных, какие данные и в каком порядке нужно вводить (это ведь можно и забыть). **Во-вторых**, результаты получаются в виде чисел на экране, без всяких пояснений их смысла. Ясно, что люди между собой так не общаются.

Любую программу составлять нужно так, чтобы ее исполнение реализовывало диалог между компьютером и пользователем в понятной для человека форме.

Прежде чем начать составление программы, нужно продумать сценарий такого диалога.

Например, составим сценарий работы программы, вычисляющей сумму двух целых чисел. На экране компьютера последовательно должны появляться следующие строки (для примера предположим, что будем вводить числа 237 и 658):

Введите первое слагаемое: A = 237
Введите второе слагаемое: B = 658
A + B = 895
Пока!

Здесь курсивом записаны символы, которые выводит компьютер по программе, а прямым жирным шрифтом — символы, вводимые пользователем.

Любой вывод на экран происходит по оператору вывода, записанному в программе.

Следовательно, с помощью оператора вывода на экран выносятся не только результаты решения задачи, но и все элементы диалога со стороны компьютера.

Вот программа, которая реализует наш сценарий:

```
Program Summa;
var A, B: integer;
begin
       write ('Введите первое слагаемое: A =');
       readln(A);
       write('Введите второе слагаемое: B =');
       readln(B);
       writeln;
       writeln('A + B = ',  A+B);
       writeln('Пока!')
end.
```

В этой программе используется возможность включать в список вывода символьные строки (они заключаются в апострофы) и арифметические выражения. Выражение **A + B** сначала вычисляется, а потом полученное число выводится на экран. Конечно, для вычисления суммы можно было написать отдельный оператор присваивания, но можно сделать и так, как в этом примере.

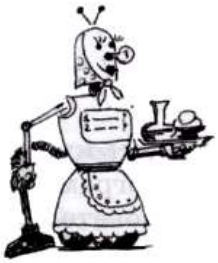
Еще обратите внимание на оператор **writeln** без списка вывода. Он обеспечивает пропуск строки на экране.

Пример программирования диалога

Компьютерная программа совсем не обязательно должна иметь математическое содержание. Вот пример сценария, судя по которому, компьютер выполняет роль электронной няньки, заботящейся о здоровье школьника. Приводятся два варианта развития сценария, в зависимости от ответа ребенка.

Вариант 1:
Ты вчера был болен. Измерь-ка температуру!
Сообщи, какая у тебя температура: 36.5
Ты здоров, дружок! Можешь идти в школу.
Желаю успехов!

Вариант 2:
Ты вчера был болен. Измерь-ка температуру!
Сообщи, какая у тебя температура: 37.3
Ты еще болен! Раздевайся и ложись в постель.
Поправляйся, дружок!



Алгоритм этой программы содержит ветвление. Идея алгоритма состоит в том, что значение температуры ребенка сравнивается с величиной нормальной температуры человека: 36,6 °С. И если у ребен ка температура выше, то он нездоров. Вот соответствующий алгоритм на АЯ:

```
алг НЯНЬКА
вещ T
нач
    вывод "Ты вчера был болен. Измерь-ка температуру!"
    вывод "Сообщи, какая у тебя температура: "
    ввод (T)
    если T> 36.6
    то вывод "Ты еще болен! Раздевайся и ложись в постель."
       вывод "Поправляйся, дружок!"
    иначе вывод "Ты здоров, дружок! Можешь идти в школу."
         вывод "Желаю успехов!"
    кв
кон
```

По этому алгоритму получается следующая программа на Паскале:

```
Program  NANNY;
Var T: real;
begin
  writeln('Ты вчера был болен. Измерь-ка
           температуру!');
  write('Сообщи, какая у тебя температура:');
  readln(T);
  if T>36.6
     then begin
             writeln('Ты еще болен! Раздевайся
                      и ложись в постель.');
             writeln('Поправляйся, дружок!')
          end
     else begin
             writeln('Ты здоров, дружок!
                      Можешь идти в школу.');
             writeln('Желаю успехов!')
          end
end.
```

Обратите внимание на два момента: **во-первых**, перед словом **else** ни в коем случае *нельзя ставить точку с запятой*; **во-вторых**, в записи и при вводе вещественных чисел целая и дробная части числа *отделяются десятичной точкой*.

Составляя подобную программу, вы сами организуете интерфейс компьютера с пользователем вашей программы. **Этот интерфейс обязательно должен быть дружественным**. Содержание диалога должно быть понятным и удобным.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§12-14, задания №6,7(§12) письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 18
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №8 Разработка программы на языке Паскаль с использованием операторов ввода, вывода, присваивания и простых ветвлений.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Разработка программы на языке Паскаль с использованием операторов ввода, вывода, присваивания и простых ветвлений.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.
2. Выполнение практического задания
Ход работы
1. Запускаем программу **PascalABC.NET**
2. Создаем новый документ и выполняем следующие задания:

а) Дано **целое** число **A**. Если значение **A > 0**, то необходимо **увеличить** его на **единицу**. Написать программу для решения этой задачи. Программа должна иметь **дружественный интерфейс**. **Протестировать** программу для **A=5, A=-4, A=0**.

б) Дано **целое** число **A**. Если значение **A < 0**, то необходимо **удвоить** его. Написать программу для решения этой задачи. Программа должна иметь **дружественный интерфейс**. **Протестировать** программу для **A=6, A=-10, A=0**.

в) Дано **целое** число **A**. Если значение **A <> 0**, то необходимо **уменьшить** его на **4**. Написать программу для решения этой задачи. Программа должна иметь **дружественный интерфейс**. **Протестировать** программу для **A=2, A=-1, A=0**.

3. Дополнительные задания:

г) Дано **целое** число **A**. Если значение **A > 0**, то необходимо **увеличить** его на **единицу**, иначе **уменьшить** на **1**. Написать программу для решения этой задачи. Программа должна иметь **дружественный интерфейс**. **Протестировать** программу для **A=3, A=0, A=-12**.

д) Дано **целое** число **A**. Если значение **A = 0**, то необходимо **увеличить** его на **3**, иначе присвоить **A** значение **равное 0**. Написать программу для решения этой задачи. Программа должна иметь **дружественный интерфейс**. **Протестировать** программу для **A=0, A=-1, A=8**.

4. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____
4. Домашнее задание
Записываем домашнее задание: §§12-14, задание №5(§14) письменно в тетради.
Комментарии учителя. *Урок окончен, до свидания!*
Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 19
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №9 Разработка программы на языке Паскаль с использованием оператора ветвления и логических операций.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Разработка программы на языке Паскаль с использованием оператора ветвления и логических операций.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **PascalABC.NET**
2. Создаем новый документ и выполняем следующие задания:
3. Разработать программы, которые**:**
    а) находит большее из двух чисел с использованием полного ветвления
    б) находит большее из двух чисел с использованием не полного ветвления
    в) находит большее из трех чисел с использованием вложенного ветвления
4. Дополнительное задание: сортирует значения двух переменных по убыванию
5. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§12-14, задание №5(§13) письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 20
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Циклы на языке Паскаль.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: Циклы на языке Паскаль.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Вы научились составлять линейные и ветвящиеся программы на Паскале. Теперь нужно освоить программирование циклов. Снова будем учиться на примере конкретной задачи. Но, в отличие от предыдущих примеров, подход к ее решению будет несколько другим.

Этапы решения расчетной задачи на компьютере

Часто задача, которую требуется решить, сформулирована не на математическом языке. Для решения на компьютере ее сначала нужно привести к форме математической задачи, а потом уже программировать.

Работа по решению таких задач с использованием компьютера проходит через следующие этапы:

1. Постановка задачи.
2. Математическая формализация.
3. Построение алгоритма.
4. Составление программы на языке программирования.
5. Отладка и тестирование программы.
6. Проведение расчетов и анализ полученных результатов.

Эту последовательность называют технологией решения задачи на компьютере.

В чистом виде программированием, т. е. разработкой алгоритма и программы, здесь являются лишь 3-й, 4-й и 5-й этапы.

На этапе постановки задачи должно быть четко определено, **что дано и что требуется найти**.

Второй этап — математическая формализация. Здесь задача переводится на язык математических формул, уравнений, отношений. Далеко не всегда эти формулы очевидны. Нередко их приходится выводить самому или отыскивать в специальной литературе. Если решение задачи требует математического описания какого-то реального объекта, явления или процесса, то формализация равносильна получению соответствующей математической модели.

Третий этап — построение алгоритма. Вы знаете два способа описания алгоритмов: блок-схемы и АЯ.

Первые три этапа — это работа без компьютера. Дальше следует собственно программирование на определенном языке в определенной системе программирования. **Последний (шестой) этап** — это использование уже разработанной программы в практических целях.

Программирование цикла на Паскале

Проследим все этапы технологии решения задачи на компьютере на примере конкретной задачи.

1. Постановка задачи. Дано N кубиков, на которых написаны разные буквы. Сколько различных N-буквенных слов можно составить из этих кубиков (слова не обязательно должны иметь смысл)?

Искомую целочисленную величину обозначим буквой F. Тогда постановка задачи выглядит так:

Дано: N.

Найти: F.

2. Математическая формализация. Получим расчетную формулу. Сначала рассмотрим несколько конкретных примеров. Имеются два кубика с буквами «И» и «К». Ясно, что из них можно составить два слова:

ИК КИ

Добавим к ним третью букву, «С». Теперь число разных слов будет в три раза больше предыдущего, т. е. равно 6:

ИКС КИС ИСК КСИ СКИ СИК

Если добавить четвертую букву, например «А», то число слов возрастет в четыре раза и станет равным 24:

КИСА КИАС КСИА КСАИ КАИС КАСИ ИКСА ИКАС ИСКА
ИСАК ИАКС ИАСК СКИА СКАИ СИКА СИАК САКИ САИК
АКИС АКСИ АИКС АИСК АСКИ АСИК.

Попробуйте записать все варианты слов из пяти букв: И, К, С, А, У. Сделать это непросто. Ясно лишь, что количество таких слов будет в пять раз больше 24, т. е. равно 120. Из шести букв можно составить 720 различных слов. С ростом числа букв число слов быстро растет. Например, для 10 букв получается 3 628 800 слов.

Подобные задачи решает раздел математики, который называется **комбинаторикой**.

Количество различных комбинаций из N предметов, получаемых изменением их порядка, называется числом **перестановок**. Это число выражается функцией от N, которая называется факториалом и записывается так:

N!

Читается: «эн факториал». Для любого натурального N значение N! вычисляется как произведение последовательности натуральных чисел от 1 до N. Например:

1! = 1;
2! = 1*2 = 2;
3! = 1*2*3 = 6;
4! = 1*2*3*4 = 24;
5! = 1*2*3*4*5 = 120
и т. д.

Теперь вернемся к формулировке задачи. Если N обозначает количество букв, а F — количество слов из этих букв, то расчетная формула такова:

F = N! = 1*2...*N.

3. Построение алгоритма. Поскольку алгоритм должен быть независимым от данного значения N, то его нельзя сделать линейным. Дело в том, что для разных N надо выполнить разное число умножений. В таком случае с изменением N линейная программа должна была бы менять длину.

Алгоритм решения данной задачи будет циклическим. С циклическими алгоритмами вы уже познакомились, работая с графическим исполнителем.

Цикл — это команда исполнителю многократно повторить указанную последовательность команд.

Рассмотрим блок-схему на рис. 2.7 и алгоритм на АЯ.

Здесь применена знакомая вам алгоритмическая структура **«цикл с предусловием»**. Выполняется она так: пока истинно условие цикла, повторяется выполнение тела цикла.

Тело цикла составляют две команды присваивания, заключенные между служебными словами нц и кц. Условие цикла — это отношение R<=N (R меньше или равно N).

В данном алгоритме переменная R выполняет роль множителя, значение которого меняется от 1 до N через единицу. Произведение накапливается в переменной F, начальное значение которой равно 1. Цикл заканчивается, когда R становится равным N + 1. Это значение в произведение уже не попадет.

```
алг СЛОВА
цел F, N, R
нач
    ввод N
    F:=1
    R:=1
    пока R<=N повторять
    нц
      F:=F*R
      R:=R+1
    кц
    вывод F
кон
```

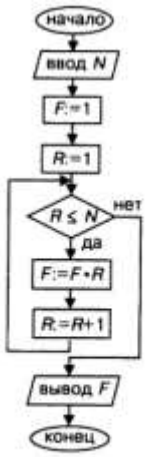


Рис. 2.7. Блок-схема алгоритма вычисления факториала

Для проверки правильности алгоритма построим трассировочную таблицу (для случая N = 3):

Шаг	Операция	N	F	R	Условие
1	ввод N	3	–	–	
2	F:=1		1	–	
3	R:=1			1	
4	R<=N				1<=3, да
5	F:=F*R		1		
6	R:=R+1			2	
7	R<=N				2<=3, да
8	F:=F*R		2		
9	R:=R+1			3	
10	R<=N				3<=3, да
11	F:=F*R		6		
12	R:=R+1			4	
13	R<=N				4<=3, нет
14	вывод F		6		
15	конец				

Из этой таблицы хорошо видно, как менялись значения переменных. Новое значение, присвоенное переменной, стирает ее старое значение (в данной таблице не повторяется запись значения переменной, если оно не изменяется; в таком виде таблица менее загромождена числами). Последнее значение F равно 6. Оно выводится в качестве результата. Очевидно, что результат верный: 3! = 6.

4. Составление программы. Чтобы составить программу решения нашей задачи, нужно научиться программировать циклы на Паскале. Основной циклической структурой является цикл с предусловием (цикл-пока). С помощью этой структуры можно построить любой циклический алгоритм. Оператор цикла с предусловием в Паскале имеет следующий формат:

while <выражение> **do** <оператор>

Служебное слово **while** означает «пока», **do** — «делать», «выполнять».

Оператор, стоящий после слова do, называется телом цикла. Тело цикла может быть простым или составным оператором, т. е. последовательностью операторов между служебными словами **begin** и **end**.

А теперь запрограммируем на Паскале алгоритм решения нашей задачи (добавив к нему организацию диалога).

```
Program Words;
var F, N, R: integer;
begin
    write('Введите число букв');
    readln(N);
    F:=1;
    R:=1;
    while R<=N  do
    begin
        F:=F*R;
        R:=R+1
    end;
    write('Из ',N,' букв можно составить ',
          F,' слов')
end.
```

Снова бросается в глаза схожесть алгоритма на АЯ и программы на Паскале. Обратите внимание на то, что в Паскале нет специальных служебных слов для обозначения конца цикла (так же как и конца ветвления). Во всех случаях, где это необходимо, используются слова **begin** и **end**.

Что такое отладка и тестирование программы

5. Отладка и тестирование. Под отладкой программы понимается процесс испытания работы программы и исправления обнаруженных при этом ошибок. Обнаружить ошибки, связанные с нарушением правил записи программы на Паскале (синтаксические и семантические ошибки), помогает используемая система програм-жирования. Пользователь получает сообщение об ошибке, исправляет ее и снова повторяет попытку исполнить программу.

Поиск алгоритмических ошибок в программе производится с помощью тестирования.

Тест — это конкретный вариант значений исходных данных, для которого известен ожидаемый результат. Прохождение теста — необходимое условие правильности программы. На тестах проверяется правильность реализации программой запланированного сценария.

Нашу программу, например, можно протестировать на значении N = 6. На экране должно получиться:

Введите число букв: 6

Из 6 букв можно составить 720 слов.

6. Проведение расчетов и анализ полученных результатов — этот этап технологической цепочки реализуется при разработке практически полезных (не учебных) программ. Например, программы «Расчет прогноза погоды». Ясно, что ею будут пользоваться длительное время, и правильность ее работы очень важна для практики. А поэтому в процессе эксплуатации эта программа может дорабатываться и совершенствоваться.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §15, вопросы №1-6 к параграфу устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 21
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №10 Разработка программ с использованием цикла с предусловием.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Разработка программ с использованием цикла с предусловием.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **PascalABC.NET**
2. Создаем новый документ и выполняем следующие задания:

а) Написать на Паскале программу, которая запрашивает с клавиатуры **N произвольных целых** чисел и ищет их **сумму**. Число **N вводится** с клавиатуры. **Разработать** для созданной программы не менее **трёх** тестов и проверить на них правильность работы алгоритма.

б) Одноклеточная амёба каждые **три часа** делится на **2** клетки. Определить, сколько амёб будет через **3, 6, 9, 12** и **24** часа.

3. Дополнительное задание: Начав тренировки, спортсмен в **первый** день пробежал **10** км. Каждый день он **увеличивал** дневную норму на **10%** нормы предыдущего дня. Какой **суммарный** путь пробежит спортсмен за **N** дней? **Протестировать** программу для **N=3** и **N=7**.
4. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §15, задание №7,8 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 22
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №11 Сочетание циклов и ветвлений.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Сочетание циклов и ветвлений.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **PascalABC.NET**
2. Создаем новый документ и выполняем следующие задания:

   а) Составить на языке Паскаль программу, которая запрашивает с клавиатуры **N целых** чисел и считает, сколько из них **положительных**, сколько **отрицательных** и сколько **нулей**. Разработать для программы не менее **двух тестов** и **доказать** правильность её работы.

3. Дополнительное задание: Написать программу, запрашивающую с клавиатуры **целые числа** и определяющую **максимальное** и **минимальное** из них. Окончание ввода – число **0**. **Протестировать** программу для последовательности: **-3 5 6 -11 24 -1 0**.
4. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §16.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 23
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Одномерные массивы в Паскале.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: Одномерные массивы в Паскале.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Изучая базы данных, электронные таблицы, вы познакомились с табличным способом организации данных. Вы уже знаете, что большие наборы данных удобно представлять в табличном виде. В таблицах могут храниться данные разных типов. На практике чаще всего приходится встречаться с таблицами, содержащими числовые и символьные (текстовые) данные.

Что такое массив

Для представления таблиц в языках программирования можно использовать массивы. Вот, например, таблица, содержащая среднемесячные значения температуры в Перми в 2000 году:

Месяц	1	2	3	4	5	6	7	8	9	10	11	12
Температура	–21	–18	–7,5	5,6	10	18	22,2	24	17	5,4	–7	–18

Такую таблицу называют **линейной**. Она представляет собой последовательность пронумерованных чисел. Для обозначения этих чисел используют **индексированные имена**. Например, через **T**[1] обозначается температура в январе (первом месяце года), **T**[5] — температура в мае и т. д.

В программировании **линейная таблица называется одномерным массивом**. В нашем примере **T** — это имя массива. Элементы массива пронумерованы. **Порядковый номер элемента называется его индексом**. Каждый элемент массива обозначается индексированным именем в следующей форме:

<имя массива> [<индекс>]

Индекс записывается в квадратных скобках: **T[2], T[10], T[12]**. Индексы могут быть представлены не только в виде констант, но и в виде целых переменных и даже выражений целого типа: **T[i], T[k], T[i+k], T[2*k]**. Важно следить, чтобы значения индексов не выходили за допустимые границы. В примере с температурами они должны лежать в диапазоне от 1 до 12.

Все элементы массива должны иметь одинаковый тип. *Если массив состоит только из целых чисел*, то **тип массива — целый**. В нашем примере **значения** температур *могут быть дробными,* поэтому **тип массива — вещественный**.

Массив — ограниченная последовательность однотипных величин.

Количество величин определяется при описании массива. Все элементы массива могут выбираться произвольно и являются одинаково доступными.

Решение задач по обработке массива связано, как правило, с перебором элементов массива. Такой перебор происходит в цикле, в котором изменяется значение индекса от начальной до конечной величины. Для того чтобы организовать ввод исходных данных в массив, нужно также использовать цикл.

Описание и ввод значений в массив в Алгоритмическом языке

Запишем алгоритм ввода значений в массив температур. Сначала посмотрим, как это делается в АЯ. Рассмотрим два варианта алгоритмов на АЯ, использующих разные способы организации цикла.

Обратите внимание на вторую строку алгоритмов. В ней присутствует описание массива температур. В АЯ массив называется таблицей. Запись

вещ таб T[1:12]

описывает таблицу (массив) вещественного типа, имя которого **T** и элементы пронумерованы от 1 до 12.

```
алг Ввод массива, вариант 1
вещ таб T[1:12]
цел I
нач
      I:=1
      пока I<=12 повторять
      нц
         вывод "T[", I , "]="
         ввод T[I]
         I:=I+1
      кц
кон
```

```
алг Ввод массива, вариант 2
вещ таб T[1:12]
цел I
нач
   для I от 1 до 12 шаг 1 повторять
   нц
      вывод "T[", I , "]="
      ввод T[I]
   кц
кон
```

Цикл с параметром в Алгоритмическом языке

В первом варианте алгоритма используется уже знакомая вам алгоритмическая структура ***цикла с предусловием***. Переменная **I** играет роль параметра цикла, изменяющегося от 1 до 12 с шагом 1. Внутри цикла она используется в качестве индекса в обозначении элементов таблицы: **T[1]**.

Ввод организован в режиме диалога. Вы уже знаете, что это обязательное условие дружественности интерфейса программы. Перед вводом каждого очередного элемента таблицы на экран будет выводиться его имя. Это результат выполнения команды вывод **"T [", I ,"]="**. После этого программист должен ввести с клавиатуры соответствующее число (команда ввод **T[I]**). На экране получим:

T[1]= -21

T[2]= -18

T[3]= -7.5 и т. д.

Во втором варианте используется алгоритмическая структура, которая называется цикл с параметром. Ее общая форма такая:

для <параметр цикла> **от** <начальное значение параметра> **до** <конечное значение параметра> **шаг** <величина приращения параметра> **повторять**

нц

<тело цикла>

кц

Параметром цикла должна быть переменная целого типа. В нашем примере это переменная I. Выполнение тела цикла повторяется для всех последовательных значений параметра от начального до конечного значения включительно с изменением его значения при каждом повторении на величину шага. Следовательно, по второму варианту алгоритма будут выполняться те же самые действия, что и по первому.

Расчет среднего значения элементов массива

Теперь сформулируем задачу обработки массива температур, которую будем решать дальше. Вычислим среднегодовую температуру. Для этого нужно сложить все 12 значений таблицы и разделить сумму на 12. Полученную величину выведем в качестве результата.

Эту задачу легко решить с помощью электронных таблиц. На рисунке 2.9 показана такая таблица. В ячейки B2:B13 заносятся значения температур. В ячейку B14 помещается формула: =СРЗНАЧ(B2:B13).

Результат (в режиме отображения значений) в ячейке B14 —2,56.

Табличный процессор — это программа, составленная программистами на некотором языке программирования. Вот мы и разберемся, как программируется вычисление среднего значения числового массива, реализованное в функции СРЗНАЧ.

	A	B
1	*Месяц*	*Температура*
2	1	–21
3	2	–18
4	3	–7,5
5	4	5,6
6	5	10
7	6	18
8	7	22,2
9	8	24
10	9	17
11	10	5,4
12	11	–7
13	12	–18
14	Среднее:	2,56

Рис. 2.9. Таблица температур

Запишем алгоритм в полном виде (с вводом, вычислениями и выводом), используя в нем для организации циклов структуру цикла с параметром.

```
алг Средняя температура
вещ таб T[1:12]
цел I, вещ Tsred
нач
    {Цикл ввода}
    для I от 1 до 12 шаг 1 повторять
    нц
        вывод "T[", I, "]="
        ввод T[I]
    кц
    {Цикл суммирования}
    Tsred:=0
    для I от 1 до 12 шаг 1 повторять
    нц
        Tsred:=Tsred+T[I]
    кц
    {Вычисление среднего}
    Tsred:=Tsred/12
    вывод "Среднегодовая температура = ", Tsred
кон
```

Обратим внимание на следующие особенности алгоритма. Появилась новая переменная **Tsred**, в которой вычисляется среднее значение:

Tsred=(T[1] + T[2] + T[3] + ... + T[12])/12.

Переменная Tsred имеет вещественный тип. Перед циклом суммирования этой переменной присваивается нулевое значение. Так всегда следует поступать с переменной, в которой накапливается сумма какой-то последовательности слагаемых. При каждом повторении цикла к значению переменной **Tsred** добавляется очередное слагаемое. После окончания цикла полученная сумма делится на 12. Это искомый результат, который выводится на экран.

А теперь посмотрим, как можно на Паскале запрограммировать алгоритм вычисления среднегодовой температуры. Для этого сначала познакомимся с правилами описания массивов. Заметим, что в этом учебнике мы ограничиваемся работой только с одномерными массивами (линейными таблицами).

Описание массива в Паскале

Формат описания одномерного массива на Паскале такой:

var <имя массива>: **array** [<Снижняя граница индекса . . верхняя граница индекса>] **of** <тип компоненты>

Слово «array» буквально переводится как «массив». Границы индекса могут быть любыми целыми числами. Важно, чтобы нижняя граница была меньше верхней границы. Описание массива температур будет следующим:

```
var T: array [1..12] of real;
```

В этой программе дважды использован оператор цикла с параметром. В простейшем случае формат этого оператора следующий:

for <параметр цикла>:=<начальное значение параметра> **to** <конечное значение параметра> **do** <тело цикла>

Значение параметра цикла увеличивается на единицу при каждом повторении тела цикла. Существует другой вариант этого оператора, в котором вместо слова to записывается **down to**. В этом случае значение параметра цикла убывает на единицу, следовательно, начальное значение должно быть больше конечного.

Так же как и для оператора цикла **while**, здесь тело цикла может быть либо простым оператором, либо составным. В первом случае тело цикла заканчивается на ближайшей точке с запятой. В нашем примере — это цикл суммирования. Во втором случае тело цикла заключается между словами **begin** и **end** (цикл ввода).

Цикл с параметром на Паскале

Рассмотрим полный текст программы на Паскале.

```
Program Temperature;
var T: array[1..12] of real;
I: integer;  Tsred: real;
begin
    {Цикл ввода}
    for I:=1 to 12 do
    begin
        write('T[',I:2,'] = ');
        readln(T[I])
    end;
    {Цикл суммирования}
    Tsred:=0;
    for I:=1 to 12 do
       Tsred:=Tsred+T[I];
    {Вычисление среднего}
    Tsred:=Tsred/12;
    writeln('Среднегодовая температура = ',
            Tsred:6:2,' градусов')
end.
```

Форматы вывода

В программе присутствует еще один новый для вас элемент Паскаля: формат вывода. Это числа с двоеточиями, стоящие после переменных в операторе вывода **write**:

write('T[',I:2,'] = ')

В этой записи 1:2 обозначает, что значение переменной I выводится как целое число в две символьные позиции на экране. Для однозначного числа в первой позиции будет помещен пробел, например:_5.

В операторе вывода результата также используется формат вывода: **Tsred:6:2**. Значение переменной Tsred выводится как смешанное число в 6 позиций, две последние из которых занимает дробная часть.

В третьей справа позиции — точка. Лишние позиции для целой части занимаются пробелами. Например: _34.25.

Результат выполнения программы **Temperature** будет выведен на экран в следующем виде:

Среднегодовая температура = 2.56 градусов

Программа с двумя массивами

	A	B
1	*Месяц*	*Температура*
2	1	–21
3	2	–18
4	3	–7,5
5	4	5,6
6	5	10
7	6	18
8	7	22,2
9	8	24
10	9	17
11	10	5,4
12	11	–7
13	12	–18
14	Среднее:	2,56

Рис. 2.9. Таблица температур

А теперь расширим условие задачи. Требуется для каждого месяца определить отклонение его средней температуры от среднегодовой величины.

Вернемся к электронной таблице на рис. 2.9. Добавим к ней еще один столбец **C**, в котором будут вычисляться искомые отклонения. В ячейку **C2** занесем формулу =B2-B14. По этой формуле вычислится отклонение январской температуры от среднегодовой. Скопировав эту формулу в ячейки **C3:C13**, получим все остальные величины. Смысл «замораживания» адреса B14 вам должен быть понятен. Результаты приведены в таблице на рис. 2.10.

	A	B	C
1	*Месяц*	*Температура*	*Отклонения*
2	1	–21	–23,56
3	2	–18	–20,56
4	3	–7,5	–10,06
5	4	5,6	3,04
6	5	10	7,44
7	6	18	15,44
8	7	22,2	19,64
9	8	24	21,44
10	9	17	14,44
11	10	5,4	2,84
12	11	–7	–9,56
13	12	–18	–20,56
14	Среднее:	2,56	

Рис. 2.10. Температуры и отклонения от среднего

Реализуем вычисление отклонений в программе на Паскале. Очевидно, в программе должен появиться еще один массив для размещения в нем таблицы отклонений. Дадим этому массиву имя Dt. Как и массив температур, он состоит из 12 чисел:

Dt[1], Dt[2], Dt[3],..., Dt[12].

К предыдущей программе надо добавить описание массива Dt в следующем виде:

var Dt: **array**[1..12] **of** real;

Значение каждого элемента массива вычисляется как разность между средней температурой в соответствующем месяце и среднегодовой температурой. Например, для января

Dt[l]:=T[1]-Tsred.

Такие вычисления повторяются в цикле 12 раз. Значения массива Dt выводятся на экран.

Запишем на Паскале фрагмент, который надо вставить в конец предыдущей программы, чтобы решить поставленную задачу.

```
for I:=1 to 12 do
begin
     Dt[I]:=T[I]-Tsred;
     writeln('Dt[',I:2,'] = ',Dt[I]:6:2)
end
```

Здесь вычисление значений массива Dt и вывод их на экран совмещены в одном цикле. Результат работы программы будет следующим:

```
Dt[ 1]= -23.56
Dt[ 2]= -20.56
Dt[ 3]= -10.06
...
Dt[12]= -20.56
```

Как и следовало ожидать, это те же самые числа, что получены в электронной таблице.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§17,18, задание №1(§18) письменно в тетради.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 24
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №12 Разработка программ обработки одномерных массивов.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Разработка программ обработки одномерных массивов.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Запускаем программу **PascalABC.NET**
2. Создаем новый документ и выполняем следующие задания:
   Разработать схему алгоритма, который вводит **массив** из **N целых чисел** и выводит на экран сам **массив** и **сумму** всех его элементов. **Протестировать** алгоритм на следующих массивах:
   а) **1 3 4 -2**
   б) **0 1 -2 10 11 12 -10 -3**
   в) **1 1 1 1 -1 -1 -1 -1**
3. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§17,18, задание №2(§18) письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 25
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №13 Разработка программы поиска числа в случайно сформированном массиве.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Разработка программы поиска числа в случайно сформированном массиве.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Отрыть учебник на странице §19 и изучить пункты параграфа: **Датчик случайных чисел на Паскале** (стр. 115) и **Алгоритм поиска числа в массиве** (стр. 116)
2. Запускаем программу **PascalABC.NET**
3. Создаем новый документ и вводим код программы **Программа поиска числа в массиве** (стр. 117)
4. Выполняем программу и проверяем результат
5. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §19, задание №1 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 26
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №14 Разработка программы на Паскале поиска минимального и максимального элементов.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Разработка программы поиска минимального и максимального элементов.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Отрыть учебник на странице §20 и изучить пункт параграфа: **Блок-схемы алгоритмов поиска максимума и минимума в массиве** (стр. 120) и **Программа на Паскале поиска максимума и минимума в массиве** (стр. 122)
2. Запускаем программу **PascalABC.NET**
3. Создаем новый документ и вводим код программы **Программа на Паскале поиска максимума и минимума в массиве** (стр. 123).
4. Выполняем программу и проверяем результат
5. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §20, задание №5 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 27
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №15 Разработка программы на Паскале сортировки массива.
Тип урока: Урок-практикум.
Цели урока:
Образовательные
Изучение вопросов: Разработка программы на Паскале сортировки массива.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Выполнение практического задания (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Выполнение практического задания

Ход работы

1. Отрыть учебник на странице §21 и изучить пункты параграфа: **Алгоритм сортировки методом пузырька** (стр. 126) и **Программа на Паскале сортировки методом пузырька** (стр. 128)
2. Запускаем программу **PascalABC.NET**
3. Создаем новый документ и вводим код программы **Программа на Паскале сортировки методом пузырька** (стр. 128)
4. Выполняем программу и проверяем результат
5. Сообщаем учителю о выполнении заданий

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§8-21, подготовка к КР №2.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 28
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Контрольная работа №2 Итоговое тестирование по теме «Программное управление работой компьютера».
Тип урока: Итоговый контроль и учет знаний и навыков.
Цели урока:
Образовательные
Изучение вопросов: итоговое тестирование по теме «Программное управление работой компьютера».
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Итоговое тестирование по теме «Программное управление работой компьютера» (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Итоговое тестирование по теме «Программное управление работой компьютера»

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§1-7, повторить.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 29
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Предыстория информатики. История ЭВМ, программного обеспечения и ИКТ.
Тип урока: Урок-лекция.
Цели урока:
Образовательные
Изучение вопросов: Предыстория информатики. История ЭВМ, программного обеспечения.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

В любой деятельности человек всегда придумывал и создавал самые разнообразные средства, приспособления, орудия труда. Все это облегчало труд, делало его производительнее, расширяло возможности людей. Известно, что история материального производства и мировой науки тесно связана с историей развития орудий труда.

Первые вспомогательные средства для работы с информацией появились много позже первых орудий материального труда. Историки утверждают, что расстояние во времени между появлением первых инструментов для физического труда (топор, ловушка для охоты) и инструментов для регистрации информационных образов (на камне, кости) составляет около миллиона лет!

Следовательно, большую часть времени существования человека на Земле труд носил только материальный характер.

Уже говорилось о том, что информационную деятельность человека можно разделить на три составляющие: хранение, передачу и обработку. Долгое время средства информационного труда развивались отдельно по этим трем направлениям.

История средств передачи информации

Первоначально люди пользовались лишь средствами ближней связи: речью, слухом, зрением. Затем появилось первое средство дальней связи — почта, чему немало способствовало развитие письменности.

Почтовый курьер инков. Такой гонец пробегал около полутора километров и передавал устное послание. Чтобы оно не задерживалось, он еще издали оповещал о своем приближении, подавая звуковые сигналы голосом или трубя в раковину.

Для быстрой передачи каких-то важных сведений часто использовались оригинальные идеи. Известно, например, применение на Кавказе **костровой связи**. Два костровых сигнальщика находились на расстоянии прямой видимости на возвышенных местах или башнях. Когда приближалась опасность (нападали враги), сигнальщики, зажигая цепочку костров, предупреждали об этом население. В XVIII веке возник **семафорный телеграф** — это тоже световая связь, но технически более совершенная.

Факельный телеграф. В Древней Греции пользовались оптической сигнальной связью: днем — дымом, ночью — огнем.

Очень богатым на открытия в области связи был XIX век. В этом веке люди овладели электричеством, которое породило множество изобретений. Сначала П. Л. Шиллинг в России в 1832 году изобрел электрический **телеграф**. А в 1837 году американец С. Морзе создал **электромагнитный телеграфный аппарат** и придумал **специальный телеграфный код — азбуку**, которая носит его имя.

В 1876 году американец А. Белл изобрел **телефон**. И наконец, в 1895 году русский изобретатель А. С. Попов открыл эпоху **радио-связи**.

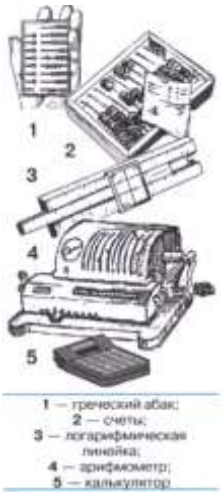
Самым замечательным изобретением XX века в области связи можно назвать **телевидение**. Освоение космоса привело к созданию **спутниковой связи**.

История средств обработки информации

Теперь познакомимся со средствами обработки информации. Важнейшим видом такой обработки являются вычисления. Появление и развитие счетных инструментов стимулировали развитие земледелия, торговли, мореплавания, астрономии и многих других областей практической и научной деятельности людей.

Нетрудно догадаться, что первым счетным средством для человека были его пальцы. Этот инструмент всегда «под рукой»! Кто из вас им не пользовался?

Вот как описывает пальцевой счет туземцев Новой Гвинеи знаменитый русский путешественник Н. Н. Миклухо-Маклай: «...папуас загибает один за другим пальцы руки, причем издает определенный звук, например "бе, бе, бе"... Досчитав до пяти, он говорит "ибон-бе" (рука). Затем он загибает пальцы другой руки, снова повторяет "бе, бе"... пока не дойдет до "ибон али" (две руки). Затем он идет дальше, приговаривая "бе, бе"... пока не дойдет до "самба-бе" и "самба-али" (одна нога, две ноги). Если нужно считать дальше, папуас пользуется пальцами рук и ног кого-нибудь другого».



1 — греческий абак;
2 — счеты;
3 — логарифмическая линейка;
4 — арифмометр;
5 — калькулятор

В V веке до нашей эры в Греции и Египте получил распространение **абак**. **«Абак»** — это греческое слово, которое переводится как «счетная доска». Вычисления на абаке производились перемещением камешков по желобам на мраморной доске.

Подобные счетные инструменты распространялись и развивались по всему миру. Например, китайский вариант абака назывался суан-пан. «Потомком» абака можно назвать и русские счеты. В России они появились на рубеже XVI-XVII веков. До недавнего времени они активно использовались, преимущественно в торговле.

В начале XVII века шотландский математик Джон Непер ввел понятие логарифма, опубликовал таблицы логарифмов. Затем в течение двух веков развивались вычислительные инструменты, основанные на использовании этой математической функции. Логарифмы позволяют свести трудоемкие арифметические операции — умножение и деление — к более простым — сложению и вычитанию. В результате появилась логарифмическая линейка. Этот инструмент до недавнего времени был вычислительным средством инженеров. И лишь ближе к концу XX столетия его вытеснили электронные калькуляторы.

В 1645 году французский математик Влез Паскаль создал первую счетную машину. Машина Паскаля позволяла быстро выполнять сложение многозначных чисел.

Немецкий ученый Лейбниц, развив идею Паскаля, создал механический арифмометр, на котором можно было выполнять все четыре арифметические операции с многозначными числами. Позднее арифмометр многократно совершенствовался, в том числе и русскими изобретателями П. Л. Чебышевым и В. Т. Однером.

Арифмометр был предшественником современного калькулятора — маленького электронно-вычислительного устройства. Сейчас практически у каждого школьника есть калькулятор, который помещается в кармане. Любому академику начала XX века такое устройство показалось бы фантастическим.

Аналитическая машина Бэббиджа — предшественница ЭВМ

Чарльз Бэббидж

Арифмометр, как и простой калькулятор, — **это средство механизации вычислений**. Человек, производя вычисления на таком устройстве, сам управляет его работой, определяет последовательность выполняемых операций. Мечтой изобретателей вычислительной техники было создание считающего автомата, который бы без вмешательства человека производил расчеты по заранее составленной программе.

Автором первого проекта вычислительного автомата был профессор Кембриджского университета Чарльз Бэббидж.

В период между 1820 и 1856 годами Бэббидж работал над созданием программно управляемой **Аналитической машины**. Это было настолько сложное механическое устройство, что проект так и не был реализован.

Можно сказать, что Бэббидж опередил свое время. Для осуществления его проекта в ту пору еще не существовало подходящей технической базы. Некоторым ученым современникам Бэббиджа его труд казался бесплодным. Однако пророчески звучат сейчас слова самого Чарльза Бэббиджа: «Природа научных знаний такова, что малопонятные и совершенно бесполезные приобретения сегодняшнего дня становятся популярной пищей для будущих поколений».

Основные идеи, заложенные в проекте Аналитической машины, в нашем веке были использованы конструкторами ЭВМ. Все главные компоненты современного компьютера присутствовали в конструкции Аналитической машины: это склад (в современной терминологии — память), где хранятся исходные числа и промежуточные результаты; мельница (арифметическое устройство), в которой осуществляются операции над числами, взятыми из склада; контора (устройство управления), производящая управление последовательностью операций над числами соответственно заданной программе; блоки ввода исходных данных и печати результатов.

Для программного управления Аналитической машиной использовались перфокарты — картонные карточки с пробитыми в них отверстиями (перфорацией). Перфокарты были изобретены в начале XIX века во Франции Жозефом М. Жаккардом для управления работой автоматического ткацкого станка.

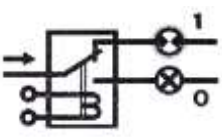
Интересным историческим фактом является то, что первую программу для машины Бэббиджа в 1846 году написала Ада Лавлейс — дочь великого английского поэта Джорджа Байрона.

Аналитическая машина Бэббиджа — это уже универсальное средство, объединяющее обработку информации, хранение информации и обмен исходными данными и результатами с человеком.

§ 22 рассказывалось о проекте Чарльза Бэббиджа — первой в истории попытке создания программно управляемого вычислительного автомата. Бэббиджу так и не удалось построить свою Аналитическую машину, используя техническую базу середины XIX столетия. К концу XIX — началу XX века с развитием электротехники появилась возможность снова вернуться к этой проблеме.

Счетно-перфорационные и релейные машины

В конце XIX века Герман Холлерит в Америке изобрел счетно-перфорационные машины. В них, так же как и в Аналитической машине Бэббиджа, использовались перфокарты, но только не для представления программы, а для хранения числовой информации. Каждая такая машина могла выполнять только одну определенную программу, манипулируя с числами, пробитыми на перфокартах. Счетно-перфорационные машины осуществляли перфорацию, сортировку, суммирование, вывод на печать числовых таблиц. На этих машинах удавалось решать многие типовые задачи статистической обработки, бухгалтерского учета и др.



Г. Холлерит основал фирму по выпуску счетно-перфорационных машин, которая затем была преобразована в фирму IBM, которая ныне является самым известным в мире производителем компьютеров.

Непосредственными предшественниками ЭВМ были релейные вычислительные машины. К 30-м годам XX века получила большое развитие релейная автоматика.

В процессе работы релейной машины происходят переключения тысяч реле из одного состояния в другое.

Электромеханическое реле — это двухпозиционный переключатель, который имеет два состояния: включено и выключено. Это свойство позволяет использовать реле для кодирования информации в двоичном виде.

Релейная машина «Марк-2», изготовленная в 1947 году, содержала около 13 ООО реле. Одной из наиболее совершенных релейных машин **была машина советского конструктора Н. И. Бессонова — РВМ-1**. Она была построена в 1956 году и проработала почти 10 лет, конкурируя с существовавшими уже в то время ЭВМ. Поскольку реле — это механическое устройство, то его

инерционность (задержка при переключении) достаточно велика, что сильно ограничивало скорость работы таких машин. Скорость РВМ-1 составляла 50 сложений или 20 умножений в секунду. Практически это был предел скорости для машин этого типа.

Начало эпохи ЭВМ

В первой половине XX века бурно развивалась радиотехника. Основным элементом радиоприемников и радиопередатчиков в то время были электронно-вакуумные лампы. Электронные лампы стали технической основой для первых **электронно-вычислительных машин (ЭВМ)**.

Первая ЭВМ — универсальная машина на электронных лампах — была построена в США в 1945 году.

Эта машина называлась **ENIAC** (расшифровывается так: ***электронный цифровой интегратор и вычислитель***). Конструкторами ENIAC были Дж. Моучли и Дж. Эккерт. Скорость счета этой машины превосходила скорость релейных машин того времени в тысячу раз.

Джон фон Нейман

Первый электронный компьютер ENIAC программировался с помощью штекерно-коммутационного способа, т. е. программа строилась путем соединения проводниками отдельных блоков машины на коммутационной доске. Эта сложная и утомительная процедура подготовки машины к работе делала ее неудобной в эксплуатации.

Основные идеи, по которым долгие годы развивалась вычислительная техника, были разработаны крупнейшим американским математиком Джоном фон Нейманом

В 1946 году в журнале Nature вышла статья Дж. фон Неймана, Г. Голдстайна и А. Беркса «Предварительное рассмотрение логической конструкции электронного вычислительного устройства». В этой статье были изложены принципы устройства и работы ЭВМ. Главный из них — принцип хранимой в памяти программы, согласно которому данные и программа помещаются в общую память машины.

Принципиальное описание устройства и работы компьютера принято называть архитектурой ЭВМ. Идеи, изложенные в упомянутой выше статье, получили название «архитектура ЭВМ Дж. фон Неймана».

В 1949 году была построена первая **ЭВМ с архитектурой Неймана — английская машина EDS AC**. Годом позже появилась **американская ЭВМ EDVAC**. Названные машины существовали в единственных экземплярах. **Серийное производство ЭВМ началось в развитых странах мира в 50-х годах XX века**.

В нашей стране первая ЭВМ была создана в 1951 году.

Называлась она **МЭСМ — малая электронная счетная машина**. Конструктором МЭСМ **был Сергей Алексеевич Лебедев**

С. А. Лебедев

Велика роль академика С. А. Лебедева в создании отечественных компьютеров. Под его руководством в 1950-х годах были построены серийные ламповые ЭВМ БЭСМ-1 (Большая электронная счетная машина), БЭСМ-2, М-20. В то время эти машины были одними из лучших в мире.

В 60-х годах XX века С. А. Лебедев руководил разработкой полупроводниковых ЭВМ БЭСМ-3М, БЭСМ-4, М-220, М-222. Выдающимся достижением того периода была машина БЭСМ-6. Это первая отечественная и одна из первых в мире ЭВМ с быстродействием один миллион операций в секунду. Последующие идеи и разработки С. А. Лебедева способствовали созданию более совершенных машин следующих поколений.

Четыре поколения ЭВМ

Электронно-вычислительную технику принято делить на поколения. Смены поколений чаще всего были связаны со сменой элементной базы ЭВМ, с прогрессом электронной техники. Это всегда приводило к росту вычислительной мощности ЭВМ, т.е. быстродействия и объема памяти. Но это не единственный признак смены поколений. При таких переходах, как правило, происходили существенные изменения в архитектуре ЭВМ, расширялся круг задач, решаемых на ЭВМ, менялся способ взаимодействия между пользователем и компьютером.

Первое поколение ЭВМ — ламповые машины 50-х годов XX века. Скорость счета самых быстрых машин первого поколения доходила до 20 тысяч операций в секунду (ЭВМ М-20). Для ввода программ и данных использовались перфоленты и перфокарты. Поскольку внутренняя память этих машин была невелика (могла вместить в себя несколько тысяч чисел и команд программы), то они, главным образом, использовались для инженерных и научных расчетов, не связанных с переработкой больших объемов данных.

Это были довольно громоздкие сооружения, содержавшие в себе тысячи ламп, занимавшие иногда сотни квадратных метров, потреблявшие электроэнергию в сотни киловатт.

Программы для таких машин составлялись на языках машинных команд. Это довольно сложная и трудоемкая работа. Программирование в те времена было доступно немногим.

В 1949 году в США был создан первый полупроводниковый прибор, заменяющий электронную лампу. Он получил название «***транзистор***». Транзисторы быстро внедрялись в радиотехнику.

В 60-х годах XX века транзисторы стали элементной базой для ЭВМ второго поколения.

Быстродействие большинства машин достигло десятков и сотен тысяч операций в секунду. Объем внутренней памяти возрос в сотни раз по сравнению с ЭВМ первого поколения.

Переход на полупроводниковые элементы улучшил качество ЭВМ по всем параметрам: они стали компактнее, надежнее, менее энергоемкими.

Большое развитие получили устройства внешней (магнитной) памяти: магнитные барабаны, накопители на магнитных лентах (НМЛ). Благодаря этому появилась возможность создавать на ЭВМ информационно-справочные, поисковые системы, позволявшие длительно хранить на магнитных носителях большие объемы информации.

Во времена второго поколения **активно стали развиваться языки программирования высокого уровня**. Первыми из них были **ФОРТРАН, АЛГОЛ, КОБОЛ**. Составление программы перестало зависеть от модели машины, сделалось проще, понятнее, доступнее.

Программирование как элемент грамотности стало широко распространяться в системе высшего образования.

Появились мониторные системы, управляющие режимом трансляции и исполнением программ. В дальнейшем из мониторных систем выросли современные операционные системы.

Третье поколение ЭВМ создавалось па новой элементной базе — интегральных схемах.

С помощью очень сложной технологии специалисты научились монтировать на маленькой пластине из полупроводникового материала, площадью менее 1 см, достаточно сложные электронные схемы. Их назвали интегральными схемами.

Первые интегральные схемы (ИС) содержали в себе десятки, затем сотни элементов (транзисторов, сопротивлений и др.). Когда степень интеграции (количество элементов) приблизилась к тысяче, их стали называть большими интегральными схемами — БИС; затем появились сверхбольшие интегральные схемы — СБИС.

ЭВМ третьего поколения начали производиться во второй половине 60-х годов прошлого века. Тогда американская **фирма IBM** приступила к выпуску системы машин IBM-360. Это были машины на ИС. Немного позднее стали выпускаться машины серии IBM-370, построенные на БИС. В Советском Союзе в 70-х годах XX века начался выпуск машин серии ЕС ЭВМ (Единая система ЭВМ) по образцу IBM-360/370.

Переход к третьему поколению связан с существенными изменениями архитектуры ЭВМ. Появилась возможность выполнять одновременно несколько программ на одной машине. Такой режим работы называется мультипрограммным (многопрограммным) режимом.

Скорость работы наиболее мощных моделей ЭВМ достигла нескольких миллионов операций в секунду. На машинах третьего поколения появился новый тип внешних запоминающих устройств — магнитные диски. Как и на магнитных лентах, на дисках можно хранить большое количество информации. Вместе с тем накопители на магнитных дисках (НМД) работают гораздо быстрее, чем НМЛ. Широко стали использоваться новые типы устройств ввода/вывода: дисплеи, графопостроители.

В этот период существенно расширились области применения ЭВМ. Стали создаваться базы данных, первые системы искусственного интеллекта, системы автоматизированного проектирования (САПР) и управления (АСУ).

В 70-е годы XX века получила мощное развитие **линия малых (мини) ЭВМ**. Своеобразным эталоном здесь стали машины американской **фирмы DEC серии PDP-11**. В нашей стране по этому образцу создавалась серия машин СМ ЭВМ (Система малых ЭВМ). Они меньше, дешевле, надежнее больших машин. Машины этого типа хорошо приспособлены для целей управления различными техническими объектами: производственными установками, лабораторным оборудованием, транспортными средствами. По этой причине их называют управляющими машинами. Во второй половине 70-х годов XX века производство мини-ЭВМ превысило производство больших машин.

Очередное революционное событие в электронике произошло **в 1971 году**, когда американская **фирма Intel объявила о создании микропроцессора**.

Микропроцессор — это сверхбольшая интегральная схема, способная выполнять функции основного блока компьютера — процессора.

Микропроцессор — это **миниатюрный «мозг», работающий по программе**. Первоначально микропроцессоры стали встраивать в различные технические устройства: станки, автомобили, самолеты. Такие микропроцессоры осуществляют автоматическое управление работой этой техники.

Соединив микропроцессор с устройствами ввода/вывода, оперативной и внешней памяти, получили новый тип компьютера: микроЭВМ.

МикроЭВМ относятся к ЭВМ четвертого поколения. Существенным отличием микроЭВМ от своих предшественников являются их малые габариты (размеры бытового телевизора) и сравнительная дешевизна. Это первый тип компьютеров, который появился в розничной продаже.

Самой популярной разновидностью ЭВМ сегодня являются персональные компьютеры.

Появление феномена персональных компьютеров (ПК) связано с именами двух американских специалистов: Стива Джобса и Стива Возняка. В 1976 году на свет появился их первый серийный ПК Apple-1, а в 1977 году — Apple-2.

Сущность того, что такое ПК, кратко можно сформулировать так:

Персональный компьютер — это микроЭВМ с дружественным к пользователю аппаратным и программным обеспечением.

В аппаратном комплекте ПК используется цветной графический дисплей, манипуляторы типа «мышь», «джойстик», удобная клавиатура, удобные для пользователя компактные диски (оптические). Программное обеспечение позволяет человеку легко общаться с машиной, быстро усваивать основные приемы работы с ней, получать пользу от компьютера, не прибегая к программированию. Общение человека и ПК может принимать форму игры с красочными картин-ками на экране, звуковым сопровождением.

Неудивительно, что машины с такими свойствами быстро приобрели популярность, причем не только среди специалистов. Персональный компьютер становится такой же привычной бытовой техникой, как радиоприемник или телевизор. Их выпускают огромными тиражами, продают в магазинах.

С 1980 года «законодателем мод» на рынке ПК становится американская фирма IBM. Ее конструкторам удалось создать такую архитектуру, которая стала фактически международным стандартом на профессиональные ПК. Машины этой серии получили название IBM PC (Personal Computer).

В конце 80-х — начале 90-х годов XX века большую популярность приобрели машины **фирмы Apple Corporation марки Macintosh**. В США они широко используются в системе образования.

Появление и распространение ПК по своему значению для общественного развития сопоставимо с появлением книгопечатания. Именно ПК сделали компьютерную грамотность массовым явлением. Развитие этого типа машин вызвало появление понятия «информационные технологии», без которых уже становится невозможным обойтись в большинстве областей деятельности человека.

Есть и другая линия в развитии ЭВМ четвертого поколения. Это ***суперкомпьютеры***. Машины этого класса имеют быстродействие в сотни миллионов и миллиарды операций в секунду. Только суперкомпьютеры могут справиться с обработкой больших объемов информации, например статистическими данными по переписи населения, результатами метеорологических наблюдений, финансовой информацией. Иногда скорость обработки информации имеет решающее значение. Примером может служить составление прогноза погоды, моделирование климатических изменений, позволяющее предсказать стихийное бедствие (цунами, тайфун, землетрясение и т. д.).

Суперкомпьютер — это многопроцессорный вычислительный комплекс. Высокое быстродействие достигается благодаря тому, что множество процессоров, его составляющих, осуществляют параллельную (одновременную) обработку данных.

Суперкомпьютеры являются дорогими машинами, стоимость которых может достигать десятков миллионов долларов. Поэтому возникает проблема доступности таких дорогих вычислительных ресурсов. Решение этой проблемы связано с созданием многопользовательских суперкомпьютерных центров.

В качестве альтернативы суперкомпьютерам создаются так называемые **кластерные системы**. ***Кластерная система — это сеть из множества рабочих станций на базе ПК***. Чтобы рабочие станции функционировали как многопроцессорная вычислительная система, в такой сети используется специальное программное обеспечение. Оказалось, что можно построить многопроцессорный

комплекс — кластер, который лишь в 2-3 раза уступает по быстродействию суперкомпьютеру, но дешевле его в сотни раз. В крупных российских университетах и научных центрах установлены и активно используются кластерные системы.

Перспективы пятого поколения

ЭВМ пятого поколения — это машины недалекого будущего. Основным их качеством должен быть высокий интеллектуальный уровень.

Машины пятого поколения — это реализованный искусственный интеллект. В них будет возможен ввод с голоса, голосовое общение, машинное «зрение», машинное «осязание». Многое уже практически сделано в этом направлении.

Вы уже хорошо знаете, что возможности современных информационных технологий определяются не только техническими характеристиками компьютеров, но и составом их программного обеспечения (ПО).

Структура программного обеспечения

Структура ПО современных персональных компьютеров схематически изображена на рис. 3.1. Именно благодаря такому разнообразию программных средств персональный компьютер оказывается полезным и школьнику, и ученому, и домохозяйке. Представление об ИКТ — информационно-коммуникационных технологиях — связано с широким распространением всего этого множества программных продуктов.

Рис. 3.1. Структура программного обеспечения компьютера

Познакомимся с историей возникновения и развития программного обеспечения ЭВМ. Появление каждого нового типа программ связано с появлением новых областей приложения компьютеров, расширением круга пользователей.

История систем программирования

Первые ЭВМ были доступны исключительно программистам. Поэтому исторически первым типом ПО стали системы программирования.

На машинах первого поколения языков программирования (в современном понимании) не существовало. Программисты работали на языке машинных кодов, что было весьма сложно. ЭВМ первого и второго поколений были приспособлены, прежде всего, для выполнения математических расчетов. А в таких расчетах часто приходится вычислять математические функции: квадратные корни, синусы, логарифмы и пр. Для вычисления этих функций программисты создавали стандартные программы, к которым производили обращения из своих расчетных программ. Стандартные программы хранились все вместе на внешнем носителе (тогда это преимущественно были магнитные ленты). Такое хранилище называлось библиотекой стандартных программ. **Библиотеки стандартных программ (БСП) — первый вид программного обеспечения ЭВМ**.

Затем в БСП стали включать стандартные программы решения типовых математических задач: вычисления корней уравнений, решения систем линейных уравнений и пр. Поскольку все эти программы носили математический характер, то в тот период чаще употреблялся термин **«математическое обеспечение ЭВМ»**. Библиотеки стандартных программ используются и в современных системах программирования (см. рис. 3.1).

В эпоху второго поколения ЭВМ распространяются **языки программирования высокого уровня (ЯПВУ)**. Об этом уже говорилось в предыдущем параграфе. ЯПВУ сделали программирование доступным не только для профессиональных программистов. Программировать стали многие научные работники, инженеры, студенты различных специальностей и даже школьники, проходящие специальную подготовку по программированию.

В программное обеспечение ЭВМ включаются **трансляторы с ЯПВУ**. Подробнее о языках программирования и трансляторах читайте в разделе 2.3 материала для дополнительного обучения. Понятие **системы программирования** в современном виде возникло в период третьего поколения ЭВМ, когда программисты для разработки программ стали пользоваться терминальным вводом (клавиатурой и дисплеем). В состав систем программирования были включены **текстовые редакторы** для ввода и редактирования программы и **отладчики**, позволяющие программисту исправлять ошибки в программе в интерактивном режиме.

История системного ПО

Операционные системы (ОС). Первые версии ОС появились еще на ЭВМ второго поколения, но массовое распространение операционные системы получают, начиная с машин третьего поколения.

Основная проблема, которую решали разработчики ОС, — повышение эффективности работы компьютера. На первых ЭВМ процессор — основное вычислительное устройство — нередко больше простаивал, чем работал во время выполнения программы. Такое происходило, если выполняемая программа часто обращалась к внешним устройствам: ввода, вывода, внешней памяти. Дело в том, что эти устройства работают в тысячи раз медленнее процессора.

Операционная система позволяет реализовать многопрограммный режим работы компьютера, при котором в состоянии выполнения находятся одновременно несколько программ. Когда одна программа обращается к внешнему устройству, процессор прерывает работу с ней (внешнее устройство продолжает работу без участия процессора) и переходит к обработке другой программы. Затем процессор может прервать работу со второй программой и продолжить выполнение первой. Таким образом, несколько программ «выстраивается в очередь» к процессору, а ОС управляет обслуживанием этой очереди. Точно так же ОС управляет обслуживанием очереди к внешним устройствам, например к принтеру. Управляют ОС и очередью к средствам ПО: трансляторам, библиотекам, прокладным программам и пр. Управление ресурсами ЭВМ — это первая функция операционных систем.

С появлением систем коллективного пользования ЭВМ операционные системы стали поддерживать **многопользовательский режим работы**. В таких системах с одной ЭВМ одновременно работают множество людей через терминальные устройства: клавиатуру и дисплей. ОС обеспечивает режим диалога с пользователями — интерактивный режим общения. При этом у каждого пользователя (программиста) создается впечатление, что он работает с компьютером один на один.

Еще одной важной функцией ОС стала организация работы с файлами. На ЭВМ третьего поколения появились магнитные диски, на которых информация хранится в файловой форме. Файловая система — это компонент ОС, работающий с файлами.

Операционные системы современных ПК также выполняют все эти функции. Особенностью, отличающей их от первых ОС, является **дружественный графический интерфейс**. А в последнее время — поддержка сетевого режима работы как в локальных, так и в глобальных сетях.

Сервисные программы. Этот тип ПО возник и начал развиваться в эпоху персональных компьютеров. Сюда входят разнообразные утилиты, антивирусные программы, программы-архиваторы.

Утилита — это небольшая программа, выполняющая действия, направленные на улучшение работы компьютера. Например, программа восстановления ошибочно удаленных файлов, программа обслуживания жесткого диска: лечения, дефрагментации и т. д.

Компьютерным вирусом является программа, способная внедряться в другие программы. Программы-вирусы выполняют нежелательные и даже опасные для работы компьютера действия: разрушают файловые структуры, «засоряют» диски и даже выводят из строя устройства компьютера. Для защиты от вирусов используются специализированные антивирусные программы (антивирус Касперского A VP, Norton Antivirus и т. д.).

Потребность в **программах-архиваторах** первоначально возникла в 80-90-х годах XX века в связи с небольшими информационными объемами устройств внешней памяти — магнитных дисков. Программа-архиватор (WinRaR, ZipMagic и др.) позволяет сократить объем файла в несколько раз без потери содержащейся в нем информации. В последнее время большое значение приобрело использование архивированных файлов в сетевых технологиях: электронной почте, файловых архивах — FTP-службе Интернета.

История прикладного ПО

Именно благодаря этому типу ПО персональные компьютеры получили широкое распространение в большинстве областей деятельности человека: медицине, экономике, образовании, делопроизводстве, торговле и даже в быту.

Самым массовым спросом среди прикладных программ пользуются, конечно, **текстовые редакторы и текстовые процессоры** (например, Microsoft Word). Ушли в прошлое пишущие машинки. Персональный компьютер, оснащенный текстовым редактором, и принтер стали основными инструментами для создания любых текстовых документов.

В 1979 году был создан первый **табличный процессор — электронная таблица VisiCalc**, ставшая самой популярной программой в среде предпринимателей, менеджеров и бухгалтеров. Идея электронной таблицы принадлежала Дэниелу Бриклину — студенту Гарвардской школы бизнеса. Начиная с 80-х годов прошлого века табличные процессоры входят в число лидирующих категорий программного обеспечения.

В конце 70-х — начале 80-х годов XX века появились первые коммерческие **системы управления базами данных (СУБД)** — программное обеспечение, которое позволяет пользователям создавать и обслуживать компьютерную базу данных, а также управлять доступом к ней. В зависимости от области применения различают:

• **настольные СУБД** (Access, FoxPro, Paradox и т. д.), предназначенные для работы с небольшими базами данных, хранящимися на локальных дисках ПК или в небольших локальных сетях;

• **СУБД серверного типа** (Oracle, SQL Server, Informix и т. д.), ориентированные на работу с большими базами данных, расположенными на компьютерах-серверах.

В настоящее время все чаще приходится обрабатывать информацию (видео, звук, анимацию), которую невозможно хранить в традиционных базах данных. Jasmine является первой в мире СУБД, ориентированной на разработку баз данных, хранящих мультимедийную информацию.

Электронный офис — в последнее время часто используемое понятие. Обычно под этим понимают такой метод ведения делопроизводства, при котором всю циркулирующую информацию обрабатывают электронным способом с помощью определенных технических средств и программного обеспечения. Таким программным обеспечением являются **интегрированные пакеты**, включающие набор приложений, каждое из которых ориентировано на выполнение определенных функций, создание документов определенного типа (текстовых документов, электронных таблиц и т. д.). В процессе работы может происходить обмен информацией между документами, могут создаваться составные документы, включающие в себя объекты разных типов (текст, рисунки, электронные таблицы).

Широко используемым сегодня интегрированным пакетом является офисная система Microsoft Office, базовыми компонентами которой принято считать текстовый редактор Microsoft Word и табличный процессор Microsoft Excel. В состав пакета также включены СУБД Microsoft Access, система подготовки презентаций Microsoft PowerPoint и некоторые другие программы. Все большей популярностью в учебных заведениях пользуются программы, входящие в свободно распространяемый пакет OpenOffice.org. Важным его достоинством является отсутствие лицензионной платы за использование. Пакет включает в себя текстовый процессор Writer, табличный процессор Calc, систему подготовки презентаций Impress, СУБД реляционного типа Base. Существуют версии OpenOffice.org, работающие в средах ОС Windows и ОС Linux.

В 90-е годы XX века появляется термин **мультимедиа**: в дополнение к традиционным тексту и графике появилась возможность работать с такими видами информации, как видео и звук. Для хранения мультимедиафайлов требуются большие объемы внешней памяти ПК, для обработки — большие процессорные мощности. Создание объемного реалистического изображения обеспечивается современными видеокартами, обработка звука — звуковой картой. Появляются программы редактирования и монтажа звука и видео, предназначенные для профессионалов в области музыки и видео. Наряду с этим создаются **программы-проигрыватели мультимедиа файлов** (Windows Media Player, Real Media Player и др.), ориентированные на широкий круг пользователей.

В 1991 году сотрудник Женевской лаборатории практической физики Тим Бернерс-Ли разрабатывает систему гипертекстовых страниц Интернета, получившую название World Wide Web (WWW) — Всемирная паутина. Создание собственной Web-страницы и опубликование ее в сети под силу многим пользователям благодаря специальным **программам-конструкторам Web-страниц**. Наиболее популярными сегодня являются Microsoft FrontPage, входящий в состав пакета Microsoft Office, и Macromedia DreamWeaver. Этими программами пользуются не только любители, но и профессионалы Web-дизайна. В ОС Linux популярна программа OpenOffice.org Write/Web.

Прикладное ПО специального назначения. Данный тип программного обеспечения служит информатизации различных профессиональных областей деятельности людей. Трудно дать исчерпывающий обзор для этой области. Сейчас практически в любой профессии, связанной с обработкой информации, существует свое специализированное ПО, свои средства информационных технологий.

Информационная технология — совокупность массовых способов и приемов накопления, передачи и обработки информации с использованием современных технических и программных средств.

ИКТ и их приложения

В последнее время в употребление вошел термин **информационно-коммуникационные технологии — ИКТ**. Рассмотрим лишь некоторые примеры профессионального использования ИКТ.

Технологии подготовки документов. Любая деловая сфера связала с подготовкой различной документации: отчетной, научной, справочной, сопроводительной, финансовой и т. д. Сегодня подготовка документа любой сложности немыслима без применения компьютера.

Для подготовки текстовых документов используются текстовые процессоры, которые прошли путь развития от **простейших редакторов**, не дающих возможность даже форматировать текст, до **текстовых процессоров**, позволяющих создавать документы, включающие в себя не только текст, но и таблицы, рисунки. Информационные технологии, связанные с созданием текстовых документов, широко используются в полиграфической промышленности. Там получили распространение **издательские системы** (например, Page Maker), позволяющие создавать макеты печатных изданий (газет, журналов, книг).

Большую роль в автоматизации подготовки финансовых документов сыграли **электронные таблицы**. С электронными таблицами работают программы — табличные процессоры. Пример такой программы — Microsoft Excel.

В настоящее время в финансовой сфере все больше используются **бухгалтерские системы** (1С-бухгалтерия и др.). Их широкое применение объясняется тем, что с помощью такой системы можно не только произвести финансовые расчеты, но и получить бумажные и электронные копии таких документов, как финансовый отчет, расчет заработной платы и пр. Электронные копии могут быть отправлены с помощью сетевых технологий в проверяющую организацию, например в налоговую инспекцию.

Для подготовки научных документов, содержащих математические расчеты, используются **математические пакеты программ** (MathCAD, Maple и пр.). Современные математические пакеты позволяют создавать документы, совмещающие текст с математическими расчетами и чертежами. С помощью такого документа можно получить результаты расчетов для разных исходных данных, изменяя их непосредственно в тексте документа. Большинство математических систем, используемых сегодня, было создано еще в середине 80-х годов прошлого столетия, т. е. вместе с появлением персональных компьютеров. Новые версии этих систем включают в себя новые возможности, например использование сетевых технологий: организацию доступа к ресурсам сети Интернет во время работы в среде математического пакета.

ИКТ в управлении предприятием. Эффективность работы компании (производственной, торговой, финансовой и пр.) зависит от того, как организованы хранение, сбор, обмен, обработка и защита информации. Для решения этих проблем уже более двадцати лет назад стали внедряться **автоматизированные системы управления (АСУ)**.

В настоящее время в этой области произошли большие перемены. Классическая АСУ включает в себя систему сбора информации, базу данных, систему обработки и анализа информации, систему формирования выходной информации. Блок обработки и анализа информации является центральным. Его работа основана на экономикоматематической модели предприятия. Он решает задачи прогнозирования деятельности компании на основе финансово-бухгалтерских расчетов, реагирования на непредвиденные ситуации, т. е. оказывает помощь в принятии управленческих решений.

Как правило, ***АСУ работают на базе локальной сети предприятия***, что обеспечивает оперативность и гибкость в принятии решений. С развитием глобальных сетей появилась **коммуникационная технология Intranet**, которую называют корпоративной паутиной. Intranet обеспечивает информационное взаимодействие между отдельными сотрудниками и подразделениями компании, а также ее отдаленными внешними партнерами. Intranet помогает поддерживать оперативную связь центрального офиса с коммерческими представительствами компании, которые обычно располагаются далеко друг от друга.

ИКТ в проектной деятельности. Информатизация произвела на свет еще одну важную технологию — **системы автоматизированного проектирования (САПР)**.

Проектирование включает в себя создание эскизов, чертежей, выполнение экономических и технических расчетов, работу с документацией.

Существуют САПРы двух видов: чертежные и специализированные. Чертежные САПРы универсальны и позволяют выполнить сложные чертежи в любой сфере технического проектирования (AutoCad). Специализированная САПР, например на проектирование жилых зданий, содержит в базе данных все необходимые сведения о строительных материалах, о стандартных строительных конструкциях, фундаментах. Инженер-проектировщик создает чертежи, производит технико-экономические расчеты с использованием таких систем. При этом повышается производительность труда конструктора, качество чертежей и расчетных работ.

Геоинформационные системы. Геоинформационные системы (ГИС) хранят данные, привязанные к географической карте местности (района, города, страны). Например, муниципальная ГИС содержит в своих базах данных информацию, необходимую для всех служб, поддерживающих жизнедеятельность города: городских властей, энергетиков, связистов, медицинских служб, милиции, пожарной службы и пр. Вся эта разнородная информация привязана к карте города. Использование ГИС помогает соответствующим службам оперативно реагировать на чрезвычайные ситуации: стихийные бедствия, экологические катастрофы, технологические аварии и пр.

ИКТ в образовании. В наше время от уровня образованности людей существенно зависит уровень развития страны, качество жизни ее населения. Требования к качеству образования постоянно растут. Старые, традиционные методы обучения уже не успевают за этими требованиями. Возникает очевидное противоречие. Использование ИКТ в образовании может помочь в разрешении этого противоречия.

Технологии обучения мало изменились за последние 100 лет. Пока в основном действует метод коллективного обучения. Не всегда такой способ обучения дает высокие результаты. Причина заключается в разном уровне способностей разных учеников. Учителя хорошо понимают, что необходим индивидуальный подход в работе с учащимися. Решению этой проблемы может помочь использование в процессе обучения специальных программ (обучающих, контролирующих, тренажерных и т. д.), входящих в состав **электронного учебника**.

Обучение — это процесс получения знаний. Традиционный источник знаний — учебник ограничен в своих информационных возможностях. Обучающимся на любой ступени образования всегда требовались дополнительные источники информации: библиотеки, музеи, архивы и пр. В этом отношении жители крупных городов находятся в более благоприятных условиях, чем сельские жители. Здесь можно говорить о существовании информационного неравенства. Решить эту проблему поможет широкое использование в обучении **информационных ресурсов Интернета**. В частности, специализированных порталов учебной информации.

Примером такого портала является портал **Единой коллекции цифровых образовательных ресурсов (ЕК ЦОР)**. Арес в Интернете: - **http://school-collection.edu.ru**. Ресурсы этого портала охватывают цифровые средства обучения по всем предметам во всех классах средней общеобразовательной школы. Сюда входят иллюстративные материалы к теории, практикумы по решению задач, ЦОРы для автоматического контроля знаний, виртуальные лаборатории и другие типы учебных программ. Особо отметим назначение **виртуальных лабораторий**. Это интерактивные программы, позволяющие воспроизводить на компьютере учебные эксперименты, чаще всего по естественнонаучным дисциплинам: физике, химии, биологии. Например, ученик может собрать на экране компьютера электрическое устройство (схему) и провести его испытание, используя виртуальные измерительные приборы: вольтметры, амперметры и др.

Еще одна проблема системы образования связана с неравными возможностями получения качественного образования из-за географической отдаленности от образовательных центров. Например, для жителя Якутии проблематично получить диплом престижного московского вуза. В решении этой проблемы на помощь приходит новая форма обучения — **дистанционное образование**, реализация которого стала возможна благодаря развитию компьютерных сетей.

Дистанционное образование приходит на смену старой форме заочного образования, при которой весь информационный обмен происходил в письменном виде через почтовую связь. Сетевое дистанционное образование позволяет вести обучение в режиме реального времени. Обучаемые могут не только читать учебный материал, но и видеть и слышать лекции крупных ученых, сдавать экзамены в прямом контакте с экзаменатором.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§22-24, вопросы к параграфам устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 30
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Социальная информатика: информационные ресурсы и информационное общество.
Тип урока: Урок-лекция.
Цели урока:
Образовательные
Изучение вопросов: Социальная информатика: информационные ресурсы и информационное общество.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Понятие информационных ресурсов

Ресурс — это запас или источник некоторых средств. Всякое общество имеет определенные ресурсы, необходимые для его жизнедеятельности. Традиционными видами общественных ресурсов являются материальные, сырьевые (природные), энергетические, трудовые, финансовые ресурсы. В дополнение к этому, одним из важнейших видов ресурсов современного общества являются **информационные ресурсы**. К информационным ресурсам уместно отнести все научно-технические знания, произведения литературы и искусства.

Информационные ресурсы — это знания, идеи человечества и указания по их реализации, зафиксированные в любой форме, на любом носителе информации.

Между информационными ресурсами и всякими иными существует важное различие: **всякий ресурс, кроме информационного, после его использования исчезает** (например, иссякают нефтяные месторождения, расходуются финансовые ресурсы). Информационным ресурсом можно пользоваться многократно. Кроме того, использование информационных ресурсов влечет за собой создание новых ресурсов, в том числе информационных. Информационные ресурсы тем быстрее растут, чем быстрее их расходуют.

Национальные информационные ресурсы

Применительно к отдельному государству говорят о **национальных информационных ресурсах**, включающих библиотечные и архивные ресурсы, научно-техническую информацию, отраслевую информацию, информацию государственных (властных) структур, информационные ресурсы социальной сферы и пр.

Виды национальных информационных ресурсов

Огромные информационные ресурсы скрыты в библиотеках. Библиотечная сеть России насчитывает 150 тысяч библиотек. Библиографическая регистрация и статистический учет всей выходящей в стране печатной продукции осуществляются Российской книжной палатой. Библиотечное дело в массовом порядке переводится на использование информационных технологий: создаются электронные каталоги, специализированные (библиографические) базы данных. В ряде центральных библиотек сформированы собственные электронные информационные ресурсы. Начался процесс публикации этих ресурсов через Интернет.

Архивы хранят материалы, связанные с историей и культурой страны. Объемы архивных материалов огромны и накапливаются зачастую быстрее, чем их удается обрабатывать. Архивный

фонд Российской Федерации ежегодно пополняется на 1,6 млн документов. В государственных архивах создано и поддерживается более 400 баз данных.

Во всех развитых странах существуют **специализированные центры научно-технической информации**, включающие специализированные издания, патентные службы и пр. Примером такого центра в нашей стране является ВИНИТИ РАН — Всероссийский институт научной и технической информации Российской академии наук. Этот информационный центр генерирует 60-70% всей научно-технической информации в России. Информация такого рода часто является дорогостоящим товаром.

Свои **отраслевые информационные ресурсы** имеются у любой промышленной, социальной и иной сферы общества.

Информационные ресурсы социальной сферы связаны с образованием, медициной, системой пенсионного обеспечения, службами занятости, медицинского и социального страхования. Например, основу информационных ресурсов в области образования составляют библиотеки вузов — высших учебных заведений (университетов, институтов, академий), общий фонд этих библиотек составляет более 300 млн экземпляров. Создана и функционирует федеральная телекоммуникационная университетская сеть. В сети Интернет отражена информация о большинстве российских вузов.

Следует отметить, что сегодня информационные ресурсы по своей значимости сопоставимы с другими ресурсами (материальными, сырьевыми, энергетическими, финансовыми). Об этом свидетельствует тот факт, что они становятся товаром, стоимость которого сопоставима со стоимостью других ресурсов. Для развитых стран характерно увеличение объема потребления из-за рубежа природных ресурсов (газ, нефть и пр.) в обмен на экспорт своих информационных ресурсов (научно-технических знаний, информационных технологий).

Цифровые информационные ресурсы. Практически все вышеперечисленные виды национальных информационных ресурсов всё в большей степени переводятся в компьютерную (цифровую) форму хранения и обработки. Поскольку не существует ограничения на цифровое представление для любых видов информации (текстов, изображений, живой речи, научно-технической документации, музыки, кино и пр.), то постепенно цифровая форма станет основной формой хранения информации. Использование же сетевых технологий делает доступ к этой информации быстрым и удобным. Основные проблемы, которые требуют решения в этих условиях: разграничение доступа, обеспечение безопасности конфиденциальной информации, соблюдение авторских прав, оплата коммерческих ресурсов.

Что такое информационное обществ

Человеческое общество вступает в период своего развития, который называют информационным обществом.

В информационном обществе преимущественным видом трудовой деятельности людей станет информационная деятельность. Информационные ресурсы становятся важнейшими из всех видов ресурсов, влияющими на общественный прогресс.

Средствами информационной деятельности людей выступает компьютерная техника, информационно-коммуникационные технологии (ИКТ).

Что такое информатизация

Обсудим еще один часто употребляемый термин: информатизация. В узком смысле **информатизация** — это процесс создания, развития и массового применения информационных средств и технологий. Этот процесс происходил в течение всей истории человеческого общества и начался отнюдь не с появлением ЭВМ. Мы уже обсуждали этот вопрос в данной главе. Именно способность работать с информацией, создавать орудия информационного труда окончательно выделила человека из мира животных.

Однако в наше время смысл и значение процесса информатизации существенно изменились. Как никогда раньше возросла роль информационных ресурсов (об этом уже говорилось ранее) для всего общества и для каждого отдельного человека.

Информатизация — организационный социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей и реализации прав граждан, органов государственной власти, органов местного самоуправления, организаций, общественных объединений на основе формирования и использования информационных ресурсов[1].

[1] Из Федерального закона «Об информации, информатизации и защите информации» от 20.02.1995 № 24

Задачи информатизации

1. ***Информационное обеспечение всех видов человеческой деятельности***. В качестве примеров видов деятельности можно назвать науку, образование, промышленное производство, сельское хозяйство, здравоохранение и многое другое. Информационное обеспечение складывается из специализированных фондов (баз данных, архивов, библиотек) и совокупности методов и средств их организации и использования.

2. ***Информационное обеспечение активного отдыха и досуга людей***. В первую очередь эта задача достигается путем обеспечения населению возможности теледоступа ко всей сокровищнице мировой культуры, а также создания индустрии телеразвлечений.

3. ***Формирование и развитие информационных потребностей людей***. Эта задача носит определенный воспитательный характер и является одной из задач общего образования. Учебная или производственная деятельность людей должна стимулировать их к повышению своего уровня информированности. Без решения этой задачи нельзя рассчитывать на успех информатизации, так как ее результаты могут оказаться просто невостребованными.

4. ***Формирование условий, обеспечивающих осуществление информатизации***. К таким условиям относятся различные виды обеспечения информатизации: экономические, организационные, научно-технические, правовые. Создание этих условий — функция государства, органов, управляющих процессами информатизации.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§25,26, вопросы к параграфам устно.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 31
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Информационная безопасность.
Тип урока: Урок-лекция.
Цели урока:
Образовательные
Изучение вопросов: Информационная безопасность.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Объяснение нового материала (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Объяснение нового материала

Информационные преступления и информационная безопасность

Многие черты информационного общества уже присутствуют в современной жизни развитых стран. Компьютеры контролируют работу атомных реакторов, распределяют электроэнергию, управляют самолетами и космическими кораблями, определяют надежность систем обороны страны и банковских систем, т. е. используются в областях общественной жизни, обеспечивающих благополучие и даже жизнь множества людей.

Жизненно важной для общества становится проблема информационной безопасности действующих систем хранения, передачи и обработки информации.

О важности этой проблемы свидетельствуют многочисленные факты. Каждые двадцать секунд в США происходит преступление с использованием программных средств. Более 80% компьютерных преступлений осуществляется через глобальную сеть Интернет, которая обеспечивает широкие возможности злоумышленникам для нарушений в глобальном масштабе. Если компьютер, являющийся объектом атаки, подключен к Интернету, то для «взломщика» нет большой разницы, где он находится — в соседней комнате или на другом континенте. Потери от хищения или повреждения компьютерных данных составляют более 100 млн долларов в год. Во многих случаях организации даже не подозревают, что «вторжение» имело место, хищение информации происходит незаметно.

Перечислим некоторые виды компьютерных преступлений, когда компьютер является инструментом для совершения преступления, а объектом преступления является информация.

1. ***Несанкционированный (неправомерный) доступ к информации***. Лицо получает доступ к секретной информации, например, путем подбора шифра (пароля). Подавляющее большинство разработок в области информационной безопасности посвящено предотвращению именно этого вида преступлений, что позволяет обеспечить охрану государственных и военных секретов.

2. ***Нарушение работоспособности компьютерной системы***. В результате преднамеренных действий ресурсы вычислительной системы становятся недоступными или снижается ее работоспособность. Примером такого рода преступлений является создание и распространение компьютерных вирусов.

3. ***Подделка (искажение или изменение), т. е. нарушение целостности компьютерной информации***. Эта деятельность является разновидностью неправомерного доступа к информации. К подобного рода действиям можно отнести подтасовку результатов голосования на выборах, референдумах и т. д. путем внесения изменений в итоговые протоколы.

Программно-технические способы защиты информации

Если речь идет о персональной информации отдельного пользователя ПК, то главной опасностью является потеря данных по непреднамеренным причинам, а также из-за случайного

проникновения вредоносных вирусов. Основные правила безопасности, которые следует соблюдать, такие:

• периодически осуществлять **резервное копирование**: файлы с наиболее важными данными дублировать и сохранять на внешних носителях;
• регулярно осуществлять **антивирусную проверку** компьютера;
• подключать ПК к электросети через **блок бесперебойного питания (ББП)**.

Блок бесперебойного питания физически защищает компьютер от последствий отключения электроэнергии или резких скачков напряжения в сети. Если компьютер от этого не защищен, то можно не только потерять данные, но и сам компьютер: какие-то его части могут выйти из строя.

Компьютерный вирус — вредоносная программа, распространяемая без ведома пользователя и наносящая тот или иной ущерб данным пользователя. Основным разносчиком вирусов является не лицензионное программное обеспечение, файлы, скопированные из случайных источников, а также службы Интернета: электронная почта, Всемирная паутина — WWW.

Каждый день в мире появляются сотни новых компьютерных вирусов. Борьбой с этим злом занимаются специалисты, создающие **антивирусные программы**. В составе программного обеспечения компьютера обязательно должны присутствовать такие программы.

Лицензионные антивирусные программы следует покупать у фирм-производителей. Однако антивирусную программу недостаточно лишь однажды установить на компьютер. После этого нужно регулярно обновлять ее базу — добавлять настройки на новые типы вирусов. Наиболее оперативно такое обновление производится через Интернет серверами фирм-производителей этих программ.

Наибольшим опасностям подвергаются пользователи глобальных сетей, Интернета. Для защиты компьютеров, подключенных к сети, от подозрительных объектов, «кочующих» по ней, используются защитные программы, которые называются «брандмауэры», (английское название firewall — противопожарная стена, огнеупор). Существуют брандмауэры, защищающие сети, подключенные к другим сетям, они называются **межсетевыми экранами**. Межсетевые экраны осуществляют фильтрацию пакетов данных, передаваемых по каналам межсетевой связи в соответствии с действующими разрешениями и запретами. Для защиты отдельных компьютеров, подключенных к сети, используются персональные брандмауэры. Брандмауэры используются для защиты от так называемых хакер- ских атак. Во всех современных операционных системах есть программы, выполняющие функции брандмауэров.

Утечка информации может происходить путем перехвата ее в процессе передачи по каналам связи. Если от этого не удается защититься техническими средствами, то применяют системы шифрования. Методами шифрования занимается **криптография**.

Правовая защита информации

Наряду с программно-техническими средствами защиты информации действуют правовые, юридические меры защиты. В 1996 году в **Уголовный кодекс РФ** был впервые внесен раздел «Преступления в сфере компьютерной информации». Он определил меру наказания за некоторые виды преступлений, ставших, к сожалению, распространенными:

• неправомерный доступ к компьютерной информации;
• создание, использование и распространение вредоносных программ для ЭВМ;
• умышленное нарушение правил эксплуатации ЭВМ и их сетей.

К защите информации относится также и осуществление авторских и имущественных прав на интеллектуальную собственность, каковой является программное обеспечение.

Необходимой составляющей общей культуры современного человека становится информационная культура. Это понятие включает в себя не только умение использовать средства информационно-коммуникационных технологий, но и соблюдение правовых норм в своей информационной деятельности.

Опасности социальных сетей

Социальной сетью называют интерактивный многопользовательский Web-сайт, контент (содержание) которого наполняется самими участниками сети. Сайт предоставляет удобную автоматизированную социальную среду, позволяющую общаться группам пользователей, имеющих общие интересы. Среди социальных сетей популярными в нашей стране являются «Одноклассники», «В контакте», «Facebook», «Twitter» и др. Переписка в социальных сетях имеет не только положительную сторону, связанную с удобством и широким кругом общения, но и свои, весьма серьезные, опасности. Главный принцип, который должны усвоить пользователи социальных сетей, состоит в следующем: размещаемая в сети деловая или личная информация навсегда сохраняется в Интернете и становится общедоступной.

Такая информация может быть найдена и использована кем угодно, в том числе и не обязательно с благими намерениями.

Всё большее число детей школьного возраста пользуется социальными сетями. Им следует помнить, что излишняя «болтливость» в Интернете может стать причиной бед лично для них и их близких. Не следует разглашать семейную информацию: о материальном достатке, о деятельности своих родителей, о времени и местах нахождения близких людей, о семейных связях и пр., т. е. всё то, что может быть использовано в преступных целях.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §22-27, подготовка к КР №3.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 32
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Контрольная работа №3 Итоговое тестирование по теме «Информационные технологии и общество».
Тип урока: Итоговый контроль и учет знаний и навыков.
Цели урока:
Образовательные
Изучение вопросов: итоговое тестирование по теме «Информационные технологии и общество».
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Итоговое тестирование по теме «Информационные технологии и общество» (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Итоговое тестирование по теме «Информационные технологии и общество»

Запустить программу «Итоговое тестирование по теме «Информационные технологии и общество»

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§1-27, подготовка к итоговой КР №4.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 33
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Контрольная работа №4 Итоговое тестирование по курсу 9 класса.
Тип урока: Итоговый контроль и учет знаний и навыков.
Цели урока:
Образовательные
Изучение вопросов: Итоговое тестирование по курсу 9 класса.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Итоговое тестирование по курсу 9 класса (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Итоговое тестирование по курсу 9 класса

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: Повторение материала за учебный год.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.

Класс: 9
Дата: «___» _____ 20__ г.
Номер урока: 34
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Повторение материала 9 класса.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные
Изучение вопросов: Повторение материала 9 класса.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Повторение материала 9 класса (32 мин.)
3. Подведение итогов урока (5 мин.)

Приемы, используемые на уроке: фронтальная работа.
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Повторение материала 9 класса

3. Подведение итогов урока
Оценки за урок, следующие _____

Урок окончен, до свидания!

Список литературы:
1. Информатика: учебник для 9 класса / И.Г. Семакин, Л.А. Залогова, С.В. Русаков, Л.В. Шестакова. – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 200 с.: ил.