ГБОУ РК «Феодосийская санаторная школа-интернат»

План-конспект уроков
по Информатике и ИКТ в 10 классе

Разработал учитель:
Калачиков Александр Владимирович

Феодосия - 2018

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 1
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Введение. Структура информатики. Техника безопасности и организация рабочего места.
Тип урока: Урок-лекция
Цель урока: рассмотреть понятие предмета информатики, ее структуры.
Задачи:

- образовательная — выработать умения различать виды информации и способы восприятия информации человеком; сформировать умение определять формы представления информации;
- развивающая — формирование представление об основных идеях и методах представления информации; демонстрация роли информации в познании действительности способствовать расширению кругозора учащихся; формирование интереса к предмету; развитие памяти и мышления;
- воспитательная — воспитание информационной культуры учащихся; воспитание самостоятельности, дисциплинированности, внимательности.

План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 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

ЗАПОМНИТЕ! К каждому рабочему месту подведено опасное для жизни напряжение.
Во время работы следует быть предельно внимательным.
Во избежание несчастного случая, поражения электрическим током, поломки оборудования рекомендуется выполнять следующие правила:
• Входите в компьютерный класс спокойно, не торопясь, не толкаясь, не задевая мебель и оборудование и только с разрешения преподавателя.
• Не включайте и не выключайте компьютеры без разрешения преподавателя.
• Не трогайте питающие провода и разъёмы соединительных кабелей.
• Не прикасайтесь к экрану и тыльной стороне монитора.
• Не размещайте на рабочем месте посторонние предметы.
• Не вставайте со своих мест, когда в кабинет входят посетители.
• Не пытайтесь самостоятельно устранять неисправности в работе аппаратуры; при неполадках и сбоях в работе компьютера немедленно прекратите работу и сообщите об этом преподавателю.
• Работайте на клавиатуре чистыми, сухими руками; легко нажимайте на клавиши, не допуская резких ударов и не задерживая клавиши в нажатом положении.

ЗАПОМНИТЕ! Если не принимать мер предосторожности, работа за компьютером может оказаться вредной для здоровья.
Чтобы не навредить своему здоровью, необходимо соблюдать ряд простых рекомендаций:
• Неправильная посадка за компьютером может стать причиной боли в плечах и пояснице. Поэтому садитесь свободно, без напряжения, не сутулясь, не наклоняясь и не наваливаясь на спинку стула. Ноги ставьте прямо на пол, одна возле другой, но вытягивайте их и не подгибайте.
• Если стул с регулируемой высотой, то её следует отрегулировать так, чтобы угол между плечом и предплечьем был чуть больше прямого. Туловище должно находиться от стола на расстоянии 15-16 см. Линия взора должна быть направлена в центр экрана. Если вы имеете очки для постоянного

ношения, работайте в очках.
• Плечи при работе должны быть расслаблены, локти — слегка касаться туловища. Предплечья должны находиться на той же высоте, что и клавиатура.
• При напряжённой длительной работе глаза переутомляются, поэтому каждые 5 минут отрывайте взгляд от экрана и смотрите на что-нибудь, находящееся вдали.

Самое главное

1. При работе за компьютером необходимо помнить: к каждому рабочему месту подведено опасное для жизни напряжение. Поэтому во время работы надо быть предельно внимательным и соблюдать все требования техники безопасности.

2. Чтобы работа за компьютером не оказалась вредной для здоровья, необходимо принимать меры предосторожности и следить за правильной организацией своего рабочего места.

Термин «информатика» может употребляться в двух смыслах:
• информатика как научная область, предметом изучения которой являются информация и информационные процессы; в которой осуществляется изобретение и создание новых средств работы с информацией;
• информатика как практическая область деятельности людей, связанная с применением компьютеров для работы с информацией.

Как современная техника немыслима без открытий теоретической физики, так и развитие информатики и информационных технологий невозможно без теории информации, теории алгоритмов и целого ряда других теорий в области кибернетики, лингвистики, семиотики, системологии и прочих наук.

В соответствии с современным пониманием, ***в информатике можно выделить четыре части:***

1) теоретическая информатика;
2) средства информатизации;
3) информационные технологии;
4) социальная информатика.

Теоретическая информатика — это научная область, предмет изучения которой — информация и информационные процессы. Как любая фундаментальная наука, теоретическая информатика раскрывает законы и принципы в своей предметной области.

Вторую и третью части в совокупности можно назвать прикладной информатикой.

Прикладная информатика — это область практического применения понятий, законов и принципов, выработанных теоретической информатикой. Прикладная информатика, безусловно, связана с применением компьютеров и информационных технологий.

В наше время таких прикладных областей очень много: это решение научных задач с помощью компьютера, издательская деятельность, разработка информационных систем, управление различными объектами и системами, техническое проектирование, компьютерное обучение, сетевые информационные технологии и многое-многое другое.

В последние годы в информатике сформировалось новое направление, которое называют **социальной информатикой.**

Его появление связано с тем, что широкое внедрение в жизнь компьютерных технологий и современных средств информационных коммуникаций (Интернета, сотовой связи) оказывает все более сильное влияние на общество в целом и на каждого отдельного человека. Общественное развитие движется к своей новой ступени — к информационному обществу.

Предметная область современной информатики очень велика и разнообразна. Как известно, нельзя объять необъятное. И наш курс затронет лишь часть тем и задач информатики. Вопросы, которые мы с вами будем изучать, относятся **к четырем важнейшим понятиям информатики:**

1) информационные процессы;
2) информационные системы;
3) информационные модели;
4) информационные технологии.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: введение, стр. 5-10.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 2
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Понятие информации.
Тип урока: Урок-лекция
Цель урока: выявить свойства информации; классифицировать информацию по видам и формам, научить учащихся выбирать понятные формы представления информации
Задачи:

- образовательная — выработать умения различать виды информации и способы восприятия информации человеком; сформировать умение определять формы представления информации;
- развивающая — формирование представление об основных идеях и методах представления информации; демонстрация роли информации в познании действительности способствовать расширению кругозора учащихся; формирование интереса к предмету; развитие памяти и мышления;
- воспитательная — воспитание информационной культуры учащихся; воспитание самостоятельности, дисциплинированности, внимательности.

План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 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
Показ видеоурока с комментариями (01. Понятие информации)

Наверное, самый сложный вопрос в информатике — это **«Что такое информация?»**. На него нет однозначного ответа. Смысл этого понятия зависит от контекста (содержания разговора, текста), в котором оно употребляется.

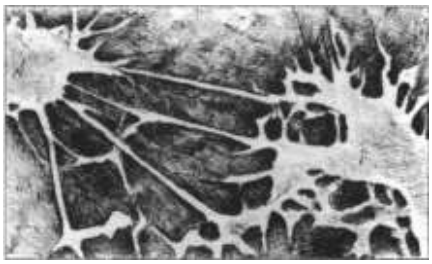
В курсе информатики основной школы информация рассматривалась в разных контекстах. С позиции человека, информация — это содержание сообщений, это самые разнообразные сведения, которые человек получает из окружающего мира через свои органы чувств. Из совокупности получаемой человеком информации формируются его знания об окружающем мире и о себе самом.

Рассказывая о компьютере, мы говорили, что ***компьютер — это универсальный программно управляемый автомат для работы с информацией***. В таком контексте не обсуждается смысл информации. Смысл — это значение, которое придает информации человек. Компьютер же работает с битами, с двоичными кодами. Вникать в их «смысл» компьютер не в состоянии. Поэтому правильнее называть информацию, циркулирующую в устройствах компьютера, данными. Тем не менее в разговорной речи, в литературе часто говорят о том, что компьютер хранит, обрабатывает, передает и принимает информацию. Ничего страшного в этом нет. Надо лишь понимать, что в «компьютерном контексте» понятие «информация» отождествляется с понятием «данные».

В Толковом словаре В. И. Даля **нет слова «информация»**. Термин «информация» начал широко употребляться с середины XX века.

В наибольшей степени понятие информации обязано своим распространением двум научным направлениям: теории связи и кибернетике. Автор теории связи Клод Шеннон, анализируя технические системы связи: телеграф, телефон, радио, рассматривал их как системы передачи информации. В таких системах информация передается в виде последовательностей сигналов: электрических или электромагнитных. Развитие теории связи послужило созданию теории информации, решающей проблему измерения информации.

Основатель кибернетики Норберт Винер анализировал разнообразные процессы управления в живых организмах и в технических системах. Процессы управления рассматриваются в кибернетике как информационные процессы. Информация в системах управления циркулирует в виде сигналов, передаваемых по информационным каналам.

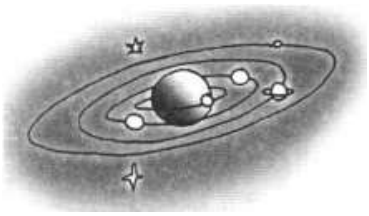


В XX веке понятие информации повсеместно проникает в науку.

Нейрофизиология (раздел биологии) изучает механизмы нервной деятельности животного и человека. Эта наука строит модель информационных процессов, происходящих в организме. Поступающая извне информация превращается в сигналы электрохимической природы, которые от органов чувств передаются по нервным волокнам к нейронам (нервным клеткам) мозга. Мозг передает управляющую информацию в виде сигналов той же природы к мышечным тканям, управляя, таким образом, органами движения. Описанный механизм хорошо согласуется с кибернетической моделью Н. Винера.

В другой биологической науке — **генетике** используется понятие наследственной информации, заложенной в структуре молекул ДНК, присутствующих в ядрах клеток живых организмов (растений, животных). Генетика доказала, что эта структура является своеобразным кодом, определяющим функционирование всего организма: его рост, развитие, патологии и пр. Через молекулы ДНК происходит передача наследственной информации от поколения к поколению.

Понятие информации относится к числу **фундаментальных**, т. е. является основополагающим для науки и не объясняется через другие понятия. В этом смысле информация встает в один ряд с такими фундаментальными научными понятиями, как вещество, энергия, пространство, время. Осмыслением информации как фундаментального понятия занимается наука философия.



Согласно одной из философских концепций, информация является свойством всего сущего, всех материальных объектов мира. Такая концепция информации называется атрибутивной (информация — атрибут всех материальных объектов). **Информация в мире возникла вместе со Вселенной**. С такой предельно широкой точки зрения, информация проявляется в воздействии одних объектов на другие, в изменениях, к которым такие воздействия приводят.

Другую философскую концепцию информации называют функциональной. Согласно функциональному подходу, **информация появилась лишь с возникновением жизни**, так как связана с функционированием сложных самоорганизующихся систем, к которым относятся живые организмы и человеческое общество. Можно еще сказать так: информация — это атрибут, свойственный только живой природе. Это один из существенных признаков, отделяющих в природе живое от неживого.

Третья философская концепция информации — антропоцентрическая, согласно которой **информация существует лишь в человеческом сознании, в человеческом восприятии**. Информационная деятельность присуща только человеку, происходит в социальных системах. Создавая информационную технику, человек создает инструменты для своей информационной деятельности.

Делая выбор между различными точками зрения, надо помнить, что всякая научная теория — лишь модель бесконечно сложного мира, поэтому она не может отражать его точно и в полной мере.

Можно сказать, что употребление понятия «информация» в повседневной жизни происходит в антропоцентрическом контексте. Для любого из нас естественно воспринимать информацию как сообщения, которыми обмениваются люди. Например, СМИ — средства массовой информации предназначены для распространения сообщений, новостей среди населения.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

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

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 3
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Представление информации, языки, кодирование.
Тип урока: Комбинированный урок
Цель урока:
Образовательные: познакомить учащихся с формами представления информации, сформировать понятие "кодирование", "декодирование"
Развивающие: способствовать развитию логического мышления
Воспитательные: воспитать аккуратность, информационную культуру.
Задача: показать необходимость умения кодировать и работать с информацией.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 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
Показ видеоурока с комментариями (02. Представление информации. Языки. Кодирование)

Из курса основной школы вам известно:
• Историческое развитие человека, формирование человеческого общества связано с развитием речи, с появлением и распространением языков. Язык — это знаковая система для представления и передачи информации.
• Люди сохраняют свои знания в записях на различных носителях. Благодаря этому знания передаются не только в пространстве, но и во времени — от поколения к поколению.
• Языки бывают естественные, например русский, китайский, английский, и формальные, например математическая символика, нотная грамота, языки программирования.

Письменность и кодирование информации
Под словом «кодирование» понимают процесс представления информации, удобный для ее хранения и/или передачи. Следовательно, запись текста на естественном языке можно рассматривать как способ кодирования речи с помощью графических элементов (букв, иероглифов). Записанный текст является кодом, заключающим в себе содержание речи, т. е. информацию. Процесс чтения текста — это обратный по отношению к письму процесс, при котором письменный текст преобразуется в устную речь. Чтение можно назвать декодированием письменного текста. Схематически эти два процесса изображены на рис. 1.1.

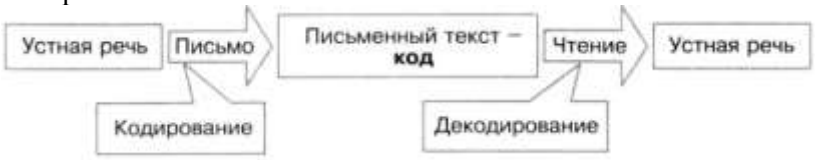


Схема на рис. 1.1 типична для всех процессов, связанных с передачей информации.

Цели и способы кодирования
Теперь обратим внимание на то, что **может существовать много способов кодирования одного и того же текста на одном и том же языке**. Например, русский текст мы привыкли записывать с помощью русского алфавита. Но то же самое можно сделать, используя латинский алфавит. Иногда так приходится поступать, отправляя SMS по мобильному телефону, на котором нет русских букв, или электронное письмо на русском языке за границу, если у адресата нет русифицированного

программного обеспечения. Например, фразу «Здравствуй, дорогой Саша!» приходится писать так: «Zdravstvui, dorogoi Sasha!».

Существует множество способов кодирования. Например, **стенография — быстрый способ записи устной речи**. Стенография появилась во времена, когда не существовало техники звукозаписи. Ею владели лишь немногие специально обученные люди — стенографисты. Они успевали записывать текст синхронно с речью выступающего человека. В стенограмме один значок обозначает целое слово или сочетание букв. Расшифровать (декодировать) стенограмму мог только сам стенографист.

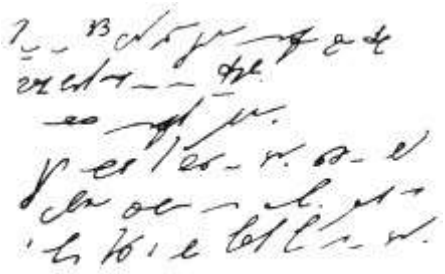


Рис. 1.2. Стенограмма

Посмотрите на текст стенограммы на рис. 1.2. Там написано следующее: «Говорить умеют все люди на свете. Даже у самых примитивных племен есть речь. Язык — это нечто всеобщее и самое человеческое, что есть на свете».

Можно придумать и другие способы кодирования.

Приведенные примеры иллюстрируют следующее важное правило: для кодирования одной и той же информации могут быть использованы разные способы; их выбор зависит от ряда обстоятельств: цели кодирования, условий, имеющихся средств.

Если надо записать текст в темпе речи, делаем это с помощью стенографии; если надо передать текст за границу, пользуемся латинским алфавитом; если надо представить текст в виде, понятном для грамотного русского человека, записываем его по правилам грамматики русского языка.

Еще одно важное обстоятельство: выбор способа кодирования информации может быть связан с предполагаемым способом ее обработки. Обсудим это на примере представления чисел — количественной информации. Используя русский алфавит, можно записать число «тридцать пять». Используя же алфавит арабской десятичной системы счисления, пишем: 35. Пусть вам надо произвести вычисления. Скажите, какая запись удобнее для выполнения расчетов: «тридцать пять умножить на сто двадцать семь» или «35 х 127»? Очевидно, что для перемножения многозначных чисел вы будете пользоваться второй записью.

Заметим, что эти две записи, эквивалентные по смыслу, используют разные языки: первая — естественный русский язык, вторая — формальный язык математики, не имеющий национальной принадлежности. Переход от представления на естественном языке к представлению на формальном языке можно также рассматривать как кодирование. Человеку удобно использовать для кодирования чисел десятичную систему счисления, а компьютеру — двоичную систему.

Широко используемыми в информатике формальными языками являются языки программирования.

В некоторых случаях возникает потребность засекречивания текста сообщения или документа, для того чтобы его не смогли прочитать те, кому не положено. Это называется защитой от несанкционированного доступа. В таком случае секретный текст шифруется. В давние времена шифрование называлось тайнописью. **Шифрование представляет собой процесс превращения открытого текста в зашифрованный, а дешифрование — процесс обратного преобразования, при котором восстанавливается исходный текст**.

Шифрование — это тоже кодирование, но с засекреченным методом, известным только источнику и адресату. Методами шифрования занимается наука криптография.

История технических способов кодирования информации

С появлением технических средств хранения и передачи информации возникли новые идеи и приемы кодирования. Первым техническим средством передачи информации на расстояние стал **телеграф, изобретенный в 1837 году** американцем Сэмюэлем Морзе. ***Телеграфное сообщение — это последовательность электрических сигналов, передаваемая от одного телеграфного аппарата по проводам к другому телеграфному аппарату.*** Эти технические обстоятельства привели Морзе к идее использования всего двух видов сигналов — короткого и длинного — для кодирования сообщения, передаваемого по линиям телеграфной связи.

Сэмюэль Финли Бриз Морзе (1791–1872), США

Такой способ кодирования получил название азбуки Морзе. В ней каждая буква алфавита кодируется последовательностью коротких сигналов (точек) и длинных сигналов (тире). Буквы отделяются друг от друга паузами — отсутствием сигналов.

В таблице на рис. 1.3 показана азбука Морзе применительно к русскому алфавиту. Специальных знаков препинания в ней нет. Их обычно записывают словами: «тчк» — точка, «зпт» — запятая и т. п.

Самым знаменитым телеграфным сообщением является сигнал бедствия «SOS» (Save Our Souls — спасите наши души). Вот как он выглядит в коде азбуки Морзе:

• • • — — — • • •

Три точки обозначают букву **S**, три тире — букву **O**. Две паузы отделяют буквы друг от друга.

А • —	И • •	Р • — •	Ш — — — —
Б — • • •	Й • — — —	С • • •	Щ — — • —
В • — —	К — • —	Т —	Ъ • — — • — •
Г — — •	Л • — • •	У • • —	Ы — • — —
Д — • •	М — —	Ф • • — •	Ь — • • —
Е •	Н — •	Х • • • •	Э • • — • •
Ж • • • —	О — — —	Ц — • — •	Ю • • — —
З — — • •	П • — — •	Ч — — — •	Я • — • —

Рис. 1.3. Кодовая таблица азбуки Морзе

Характерной особенностью азбуки Морзе является переменная длина кода разных букв, поэтому **код Морзе называют неравномерным кодом**. Буквы, которые встречаются в тексте чаще, имеют более короткий код, чем редкие буквы. Например, код буквы «Е» — одна точка, а код буквы «Ъ» состоит из шести знаков. Зачем так сделано? Чтобы сократить длину всего сообщения. Но из-за переменной длины кода букв возникает проблема отделения букв друг от друга в тексте. Поэтому приходится для разделения использовать паузу (пропуск). Следовательно, телеграфный алфавит Морзе является троичным, так как в нем используется три знака: точка, тире, пропуск.

Жан Морис Эмиль Бодо (1845–1903), Франция

Равномерный телеграфный код был изобретен французом Жаном Морисом Бодо в конце XIX века. В нем использовалось всего два вида сигналов. Неважно, как их назвать: точка и тире, плюс и минус, ноль и единица.

Это два отличающихся друг от друга электрических сигнала.

В коде Бодо длина кодов всех символов алфавита одинакова и равна пяти. В таком случае не возникает проблемы отделения букв друг от друга: каждая пятерка сигналов — это знак текста.

Код Бодо — это первый в истории техники способ двоичного кодирования информации. Благодаря идее Бодо удалось автоматизировать процесс передачи и печати букв. Был создан клавишный телеграфный аппарат. Нажатие клавиши с определенной буквой вырабатывает соответствующий пятиимпульсный сигнал, который передается по линии связи. Принимающий аппарат под воздействием этого сигнала печатает ту же букву на бумажной ленте.

Из курса информатики основной школы вам известно, что **в современных компьютерах для кодирования текстов также применяется равномерный двоичный код**. Проблемы кодирования информации в компьютере и при передаче данных по сети мы рассмотрим несколько позже.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

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

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 4
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №1 «Шифрование данных».
Тип урока: Урок-практикум.
Цель урока:
Образовательные: знакомство с простейшими приемами шифрования и дешифрования текстовой информации.
Развивающие: способствовать развитию логического мышления
Воспитательные: воспитать аккуратность, информационную культуру.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 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. Практическая работа

Ход работы

Задание 1

Шифр Цезаря. Этот шифр реализует следующее преобразование текста: каждая буква исходного текста заменяется следующей после нее буквой в алфавите, который считается написанным по кругу.

Используя шифр Цезаря, зашифровать следующие фразы:
а) Делу время - потехе час
б) С Новым годом
в) Первое сентября

Задание 2

Используя шифр Цезаря, декодировать следующие фразы:
а) Лмбттоьк шбт
б) Вёмпё тпмочё рфтуьой

Задание 3

Шифр Виженера. Это шифр Цезаря с переменной величиной сдвига. Величину сдвига задают ключевым словом. Например, ключевое слово ВАЗА означает следующую последовательность сдвигов букв исходного текста: 3 1 9 1 3 1 9 1 и т.д. Используя в качестве ключевого слово ЗИМА, закодировать слова: АЛГОРИТМИЗАЦИЯ, КОМПЬЮТЕР, ИНТЕРНЕТ.

Задание 4

Слово ЖПЮЩЕБ получено с помощью шифра Виженера с ключевым словом БАНК. Восстановить исходное слово.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: §§1,2.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 5
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Измерение информации. Алфавитный подход.
Тип урока: Комбинированный урок.
Цель урока:
Образовательные: закрепление понятия алфавита; мощности (размера) алфавита, формул для нахождения объема информации; закрепление умения решать задачи алфавитным подходом.
Развивающие: развитие логического мышления учащихся, умения сопоставлять, анализировать, делать выводы.
Воспитательные: воспитывать аккуратность, внимательность, формирование познавательного интереса к предмету.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 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
Показ видеоурока с комментариями (03. Измерение информации. Алфавитный подход)

Вопрос об измерении количества информации является очень важным как для науки, так и для практики. В самом деле, если информация является предметом нашей деятельности, мы ее храним, передаем, принимаем, обрабатываем. Поэтому важно договориться о способе ее измерения, позволяющем, например, ответить на вопросы: достаточно ли места на носителе, чтобы разместить нужную нам информацию, или сколько времени потребуется, чтобы передать ее по имеющемуся каналу связи. Величина, которая нас в этих ситуациях интересует, называется **объемом информации**. В таком случае говорят об алфавитном, или объемном, подходе к измерению информации.

Алфавитный подход к измерению информации применяется в цифровых (компьютерных) системах хранения и передачи информации. В этих системах используется двоичный способ кодирования информации. При алфавитном подходе для определения количества информации имеет значение лишь размер (объем) хранимого и передаваемого кода. **Алфавитный подход** еще называют **объемным подходом**. Из курса информатики 7-9 классов вы знаете, что если с помощью i-разрядного двоичного кода можно закодировать алфавит, состоящий из N символов (где N — целая степень двойки), то эти величины связаны между собой по формуле:

$$2^i = N.$$

Число N называется мощностью алфавита.

Если, например, **i = 2**, то можно построить 4 двухразрядные комбинации из нулей и единиц, т. е. **закодировать 4 символа**. При i = 3 существует 8 трехразрядных комбинаций нулей и единиц (кодируется 8 символов):

$i = 2$:	00	01	10	11				
$i = 3$:	000	001	010	011	100	101	110	111

Английский алфавит содержит 26 букв. Для записи текста нужны еще как минимум шесть символов: пробел, точка, запятая, вопросительный знак, восклицательный знак, тире. В сумме получается расширенный алфавит мощностью в 32 символа.

Поскольку $32 = 2^5$, все символы можно закодировать всевозможными пятиразрядными двоичными кодами от 00000 до 11111. Именно пятиразрядный код использовался в телеграфных аппаратах,

появившихся еще в XIX веке. Телеграфный аппарат при вводе переводил английский текст в двоичный код, длина которого в 5 раз больше, чем длина исходного текста.

В двоичном коде каждая двоичная цифра несет одну единицу информации, которая называется 1 бит.

Бит является основной единицей измерения информации.

Длина двоичного кода, с помощью которого кодируется символ алфавита, называется ***информационным весом символа***. В рассмотренном выше примере информационный вес символа расширенного английского алфавита оказался равным 5 битам.

Информационный объем текста складывается из информационных весов всех составляющих текст символов. Например, английский текст из 1000 символов в телеграфном сообщении будет иметь информационный объем 5000 битов.

Алфавит русского языка включает 33 буквы. Если к нему добавить еще пробел и пять знаков препинания, то получится набор из 39 символов. Для двоичного кодирования символов такого алфавита пятиразрядного кода уже недостаточно. Нужен как минимум 6-разрядный код. Поскольку $2^6 = 64$, остается еще резерв для 25 символов (64 - 39 = 25). Его можно использовать для кодирования цифр, всевозможных скобок, знаков математических операций и других символов, встречающихся в русском тексте. Следовательно, информационный вес символа в расширенном русском алфавите будет равен 6 битам. А текст из 1000 символов будет иметь объем 6000 битов.

А. Н. Колмогоров (1903–1987), Россия

Итак, если ***i*** — информационный вес символа алфавита, а ***K*** — количество символов в тексте, записанном с помощью этого алфавита, то информационный объем ***I*** текста выражается формулой:

I = ***K*** x ***i*** (битов).

Идея измерения количества информации в сообщении через длину двоичного кода этого сообщения принадлежит выдающемуся российскому математику Андрею Николаевичу Колмогорову. Согласно Колмогорову, количество информации, содержащееся в тексте, определяется минимально возможной длиной двоичного кода, необходимого для представления этого текста.

Для определения информационного веса символа полезно знать ряд целых степеней двойки. Вот как он выглядит в диапазоне от 2^1 до 2^{10}:

i	1	2	3	4	5	6	7	8	9	10
2^i	2	4	8	16	32	64	128	256	512	1024

Поскольку мощность N алфавита может не являться целой степенью двойки, информационный вес символа алфавита мощности N определяется следующим образом. Находится ближайшее к N значение во второй строке таблицы, не меньшее чем N.

Соответствующее значение ***i*** в первой строке **будет равно информационному весу символа**.

Пример. Определим информационный вес символа алфавита, включающего в себя все строчные и прописные русские буквы (66); цифры (10); знаки препинания, скобки, кавычки (10). Всего получается 86 символов.

Поскольку $2^6 < 86 < 2^7$, информационный вес символов данного алфавита равен 7 битам. Это означает, что все 86 символов можно закодировать семиразрядными двоичными кодами.

Для двоичного представления текстов в компьютере чаще всего применяется восьмиразрядный код. С помощью восьмиразрядного кода можно закодировать алфавит из 256 символов, поскольку $256 = 2^8$. В стандартную кодовую таблицу (например, используемую в ОС Windows таблицу ANSI) помещаются все необходимые символы: английские и русские буквы — прописные и строчные, цифры, знаки препинания, знаки арифметических операций, всевозможные скобки и пр.

Более крупной, чем бит, единицей измерения информации является байт: 1 байт = 8 битов.

Информационный объем текста в памяти компьютера измеряется в байтах. Он равен количеству символов в записи текста.

Одна страница текста на листе формата А4 кегля 12 с одинарным интервалом между строками в компьютерном представлении будет иметь объем 4000 байтов, так как на ней помещается примерно 4000 знаков.

Помимо бита и байта, для измерения информации используются и более крупные единицы:

1 Кб (килобайт) = 2^{10} байтов = 1024 байта;
1 Мб (мегабайт) = 2^{10} Кб = 1024 Кб;
1 Гб (гигабайт) = 2^{10} Мб = 1024 Мб;
1 Тб (терабайт) = 2^{10} Гб = 1024 Гб.

Объем той же страницы текста будет равен приблизительно 3,9 Кб. А книга из 500 таких страниц займет в памяти компьютера примерно 1,9 Мб.

В компьютере любые виды информации: тексты, числа, изображения, звуки — представляются в форме двоичного кода.

Объем информации любого вида, выраженный в битах, равен длине двоичного кода, в котором эта информация представлена.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: §3, задания№ 8-9 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 6
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Измерение информации. Содержательный подход.
Тип урока: Комбинированный урок.
Цель урока: приобретение обучающимися теоретических знаний и практических навыков по измерению количество информации с точки зрения содержательного подхода.
Задачи урока:
образовательные: • сформировать у учащихся представление о содержательном подходе к измерению информации; • познакомить с понятием «вероятность»; • научить вычислять количество информации в сообщении о некотором событии; • научить решать задачи на измерение информации;
развивающие: • развивать культуру речи, мышление (умение сравнивать, анализировать, обобщать); • учить ставить и разрешать проблемы, делать выводы;
воспитательная: • воспитывать уважительное отношение к мнению других.
План урока:
1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала
Показ видеоурока с комментариями (04. Измерение информации. Содержательный подход)

Содержательный подход к измерению информации отталкивается от определения информации как содержания сообщения, получаемого человеком. Сущность содержательного подхода заключается в следующем: сообщение, информирующее об исходе какого-то события, снимает неопределенность знания человека об этом событии.

Чем больше первоначальная неопределенность знания, тем больше информации несет сообщение, снимающее эту неопределенность.

Приведем примеры, иллюстрирующие данное утверждение.

Ситуация 1. В ваш класс назначен новый учитель информатики; на вопрос «Это мужчина или женщина?» вам ответили: «Мужчина».

Ситуация 2. На чемпионате страны по футболу играли команды «Динамо» и «Зенит». Из спортивных новостей по радио вы узнаете, что игра закончилась победой «Зенита».

Ситуация 3. На выборах мэра города было представлено четыре кандидата. После подведения итогов голосования вы узнали, что избран Н. Н. Никитин.

Вопрос: в какой из трех ситуаций полученное сообщение несет больше информации?

Неопределенность знания — это количество возможных вариантов ответа на интересовавший вас вопрос. Еще можно сказать: возможных исходов события. Здесь событие — например, выборы мэра; исход — выбор, например, Н. Н. Никитина.


Клод Элвуд Шеннон (1916–2001)

В первой ситуации 2 варианта ответа: мужчина, женщина; во второй ситуации 3 варианта: выиграл «Зенит», ничья, выиграло «Динамо»; в третьей ситуации — 4 варианта: 4 кандидата на пост мэра.

Согласно данному выше определению, наибольшее количество информации несет сообщение в третьей ситуации, поскольку неопределенность знания об исходе события в этом случае была наибольшей.

В 40-х годах XX века проблема измерения информации была решена американским ученым Клодом Шенноном — основателем теории информации.

Согласно Шеннону, **информация — это снятая неопределенность знания человека об исходе какого-то события**.

В теории информации единица измерения информации определяется следующим образом.

Сообщение, уменьшающее неопределенность знания об исходе некоторого события в два раза, несет 1 бит информации.

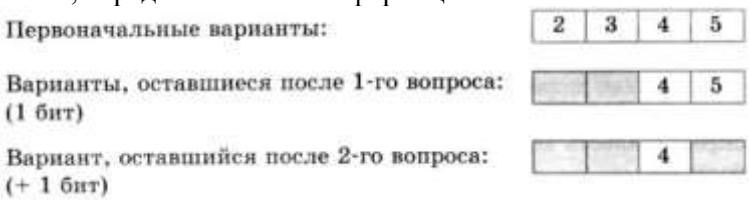
Согласно этому определению, сообщение в первой из описанных ситуаций несет 1 бит информации, поскольку из двух возможных вариантов ответа был выбран один.

Следовательно, количество информации, полученное во второй и в третьей ситуациях, больше, чем один бит. Но как измерить это количество?

Рассмотрим еще один пример.

Ученик написал контрольную по информатике и спрашивает учителя о полученной оценке. Оценка может оказаться любой: от 2 до 5. На что учитель отвечает: «Угадай оценку за два вопроса, ответом на которые может быть только "да" или "нет"». Подумав, ученик задал первый вопрос: «Оценка выше тройки?». «Да», — ответил учитель. Второй вопрос: «Это пятерка?». «Нет», — ответил учитель. Ученик понял, что он получил четверку. Какая бы ни была оценка, таким способом она будет угадана!

Первоначально неопределенность знания (количество возможных оценок) была равна четырем. С ответом на каждый вопрос неопределенность знания уменьшалась в 2 раза и, следовательно, согласно данному выше определению, передавался 1 бит информации.



Узнав оценку (одну из четырех возможных), ученик получил 2 бита информации.

Рассмотрим еще один частный пример, а затем выведем общее правило.

Вы едете на электропоезде, в котором 8 вагонов, а на вокзале вас встречает товарищ. Товарищ позвонил вам по мобильному телефону и спросил, в каком вагоне вы едете. Вы предлагаете угадать номер вагона, задав наименьшее количество вопросов, ответами на которые могут быть только слова «да» или «нет».

Немного подумав, товарищ стал спрашивать:

— Номер вагона больше четырех?
— Да.
— Номер вагона больше шести?
— Нет.
— Это шестой вагон?
— Нет.
— Ну теперь все ясно! Ты едешь в пятом вагоне!

Схематически поиск номера вагона выглядит так:

Каждый ответ уменьшал неопределенность знания в два раза. Всего было задано три вопроса. Значит, в сумме набрано 3 бита информации. То есть сообщение о том, что вы едете в пятом вагоне, несет 3 бита информации.

Способ решения проблемы, примененный в примерах с оценками и вагонами, называется методом половинного деления: ответ на каждый вопрос уменьшает неопределенность знания, имеющуюся перед ответом на этот вопрос, наполовину. Каждый такой ответ несет 1 бит информации.

Заметим, что решение подобных проблем методом половинного деления наиболее рационально. Таким способом всегда можно угадать, например, любой из восьми вариантов за 3 вопроса. Если бы поиск производился последовательным перебором: «Ты едешь в первом вагоне?» «Нет», «Во втором вагоне?» «Нет» и т. д., то про пятый вагон вы смогли бы узнать после пяти вопросов, а про восьмой — после восьми.

«Главная формула» информатики

Сформулируем одно очень важное условие, относящееся к рассмотренным примерам. Во всех ситуациях предполагается, что все возможные исходы события равновероятны. Равновероятно, что учитель может быть мужчиной или женщиной; равновероятен любой исход футбольного матча, равновероятен выбор одного из четырех кандидатов в мэры города. То же относится и к примерам с оценками и вагонами.

Тогда полученные нами результаты описываются следующими формулировками:
• сообщение об одном из двух равновероятных исходов некоторого события несет 1 бит информации;
• сообщение об одном из четырех равновероятных исходов некоторого события несет 2 бита информации;
• сообщение об одном из восьми равновероятных исходов некоторого события несет 3 бита информации.

Обозначим буквой N количество возможных исходов события, или, как мы это еще называли, — неопределенность знания. Буквой **i** будем обозначать количество информации в сообщении об одном из **N** результатов.

В примере с учителем: N = 2, i = 1 бит;
в примере с оценками: N = 4, i = 2 бита;
в примере с вагонами: N = 8, i = 3 бита.

Нетрудно заметить, что связь между этими величинами выражается следующей формулой:

$2^i = N$.

Действительно: $2^1 = 2$; $2^2 = 4$; $2^3 = 8$.

С полученной формулой вы уже знакомы из курса информатики для 7 класса и еще не однажды с ней встретитесь. Значение этой формулы столь велико, что мы назвали ее **главной формулой информатики**. Если величина N известна, а i неизвестно, то данная формула становится уравнением для определения i. В математике такое уравнение называется показательным уравнением.

Пример. Вернемся к рассмотренному выше примеру с вагонами. Пусть в поезде не 8, а 16 вагонов. Чтобы ответить на вопрос, какое количество информации содержится в сообщении о номере искомого вагона, нужно решить уравнение:

$$2^i = 16.$$

Поскольку $16 = 2^4$, то i = 4 бита.

Количество информации i, содержащееся в сообщении об одном из N равновероятных исходов некоторого события, определяется из решения показательного уравнения:

$$2^i = N.$$

Пример. В кинозале 16 рядов, в каждом ряду 32 места. Какое количество информации несет сообщение о том, что вам купили билет на 12-й ряд, 10-е место?

Решение задачи: в кинозале всего 16 • 32 = 512 мест. Сообщение о купленном билете однозначно определяет выбор одного из этих мест. Из уравнения $2^i = 512 = 2^9$ получаем: **i** - 9 битов.

Но эту же задачу можно решать иначе. Сообщение о номере ряда несет 4 бита информации, так как $2^4 = 16$. Сообщение о номере места несет 5 битов информации, так как $2^5 = 32$. В целом сообщение про ряд и место несет: 4 + 5 = 9 битов информации.

Данный пример иллюстрирует выполнение закона ***аддитивности количества информации (правило сложения): количество информации в сообщении одновременно о нескольких результатах независимых друг от друга событий равно сумме количеств информации о каждом событии отдельно***.

Сделаем одно важное замечание. С формулой $2^i = N$ мы уже встречались, обсуждая алфавитный подход к измерению информации (см. § 3. Измерение информации. Алфавитный подход). В этом случае N рассматривалось как мощность алфавита, а i — как информационный вес каждого символа алфавита. Если допустить, что все символы алфавита появляются в тексте с одинаковой частотой, т. е. равновероятно, то информационный вес символа i тождественен количеству информации в сообщении о появлении любого символа в тексте. При этом N — неопределенность знания о том, какой именно символ алфавита должен стоять в данной позиции текста. Данный факт демонстрирует связь между алфавитным и содержательным подходами к измерению информации.

Формула Хартли

Если значение N равно целой степени двойки (4, 8, 16, 32, 64 и т. д.), то показательное уравнение легко решить в уме, поскольку i будет целым числом. А чему равно количество информации в сообщении о результате матча «Динамо»-«Зенит»? В этой ситуации N = 3. Можно догадаться, что решение уравнения

$2^i = 3$.

будет дробным числом, лежащим между 1 и 2, поскольку $2^1 = 2 < 3$, а $2^2 = 4 > 3$. А как точнее узнать это число?

В математике существует функция, с помощью которой решается показательное уравнение. Эта функция называется **логарифмом**, и решение нашего уравнения записывается следующим образом:

$$i = \log_2 N.$$

Читается это так: «логарифм от N по основанию 2». Смысл очень простой: логарифм по основанию 2 от А — это степень, в которую нужно возвести 2, чтобы получить N. Например, вычисление уже известных вам значений можно представить так:

$\log_2 2 = 1$, $\log_2 4 = 2$, $\log_2 8 = 3$.

Значения логарифмов находятся с помощью специальных логарифмических таблиц. Также можно использовать инженерный калькулятор или табличный процессор. Определим количество информации, полученной из сообщения об одном исходе события из трех равновероятных, с помощью электронной таблицы. На рисунке 1.4 представлены два режима электронной таблицы: режим отображения формул и режим отображения значений.

	A	B
1	*N*	*i (битов)*
2	3	=LOG(A2;2)

	A	B
1	*N*	*i (битов)*
2	3	1,584962501

Ральф Хартли
(1888–1970)

Рис. 1.4. Определение количества информации в электронных таблицах с помощью функции логарифма

В табличном процессоре Microsoft Excel функция логарифма имеет следующий вид: LOG (аргумент; основание). Аргумент — значение N находится в ячейке А2, а основание логарифма равно 2. В результате получаем с точностью до девяти знаков после запятой: $i = \log_2 3 = 1{,}584962501$ (бита).

Формула для измерения количества информации: $i = \log_2 N$ была предложена американским ученым Ральфом Хартли — одним из основоположников теории информации.

Формула Хартли: $i = \log_2 N$

Здесь i — количество информации, содержащееся в сообщении об одном из N равновероятных исходов события.

Данный пример показал, что количество информации, определяемое с использованием содержательного подхода, может быть дробной величиной, в то время как информационный объем, вычисляемый путем применения алфавитного подхода, может иметь только целочисленное значение.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

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

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 7
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №2: «Измерение информации».
Тип урока: Урок-практикум.
Цель урока: приобретение обучающимися теоретических знаний и практических навыков по измерению количество информации.
Задачи урока:
образовательные: сформировать у учащихся представление о содержательном подходе к измерению информации; познакомить с понятием «вероятность»; научить вычислять количество информации в сообщении о некотором событии; научить решать задачи на измерение информации;
развивающие: развивать культуру речи, мышление (умение сравнивать, анализировать, обобщать); учить ставить и разрешать проблемы, делать выводы;
воспитательная: воспитывать уважительное отношение к мнению других.
План урока:

1. Организационный момент (3 мин.)
2. Практическая работа (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Ход работы

Вариант 1
1. Сообщение, записанное буквами из 128-символьного алфавита, содержит 45 символов. Какой объем информации оно несет?
2. Сколько бит составляет сообщение, содержащее 0,25 Кбайт?
3. Информационное соо бщение объемом 2,5 Кбайта содержит 2560 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 16-символьный алфавит. Каждая страница содержит 32 строк по 128 символов в строке. Какой объем информации содержат 8 страниц текста?
5. Имеется файл с текстом из 20000 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 20 Кбайт. Поместится ли текст на диск?

Вариант 2
1. Сообщение, записанное буквами из 16-символьного алфавита, содержит 130 символов. Какой объем информации оно несет?
2. Сколько Кбайт составляет сообщение, содержащее 6,5 Мбайт?
3. Информационное сообщение объемом 1,5 Кбайта содержит 3072 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 16-символьный алфавит. Каждая страница содержит 28 строк по 134 символов в строке. Какой объем информации содержат 7 страниц текста?
5. Имеется файл с текстом из 15000 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 18 Кбайт. Поместится ли текст на диск?

Вариант 3

1. Сообщение, записанное буквами из 128-символьного алфавита, содержит 120 символов. Какой объем информации оно несет?
2. Сколько Кбайт составляет сообщение, содержащее 8192 бит?
3. Информационное сообщение объемом 0,125 Кбайта содержит 256 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 4-символьный алфавит. Каждая страница содержит 30 строк по 70 символов в строке. Какой объем информации содержат 5 страниц текста?
5. Имеется файл с текстом из 17500 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 15 Кбайт. Поместится ли текст на диск?

Вариант 4

1. Сообщение, записанное буквами из 32-символьного алфавита, содержит 90 символов. Какой объем информации оно несет?
2. Сколько бит составляет сообщение, содержащее 0,5 Кбайта?
3. Информационное сообщение объемом 0,25 Кбайта содержит 1024 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 32-символьный алфавит. Каждая страница содержит 25 строк по 110 символов в строке. Какой объем информации содержат 9 страниц текста?
5. Имеется файл с текстом из 18000 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 31 Кбайт. Поместится ли текст на диск?

Вариант 5

1. Сообщение, записанное буквами из 256-символьного алфавита, содержит 45 символов. Какой объем информации оно несет?
2. Сколько Кбайт составляет сообщение, содержащее 1,5 Мбайта?
3. Информационное сообщение объемом 3 Кбайта содержит 4096 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 256-символьный алфавит. Каждая страница содержит 30 строк по 70 символов в строке. Какой объем информации содержат 5 страниц текста?
5. Имеется файл с текстом из 31000 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 12 Кбайт. Поместится ли текст на диск?

Вариант 6

1. Сообщение, записанное буквами из 64-символьного алфавита, содержит 85 символов. Какой объем информации оно несет?
2. Сколько Кбайт составляет сообщение, содержащее 16384 бит?
3. Информационное сообщение объемом 2,5 Кбайта содержит 5120 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 64-символьный алфавит. Каждая страница содержит 56 строк по 92 символов в строке. Какой объем информации содержат 4 страниц текста?
5. Имеется файл с текстом из 16250 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 24 Кбайт. Поместится ли текст на диск?

Вариант 7

1. Сообщение, записанное буквами из 32-символьного алфавита, содержит 140 символов. Какой объем информации оно несет?
2. Сколько байт составляет сообщение, содержащее 4416 бит?
3. Информационное сообщение объемом 1,5 Кбайта содержит 2048 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 128-символьный алфавит. Каждая страница содержит 42 строк по 115 символов в строке. Какой объем информации содержат 6 страниц текста?

5. Имеется файл с текстом из 22300 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 21 Кбайт. Поместится ли текст на диск?

Вариант 8

1. Сообщение, записанное буквами из 256-символьного алфавита, содержит 74 символов. Какой объем информации оно несет?
2. Сколько Кбайт составляет сообщение, содержащее 204800 бит?
3. Информационное сообщение объемом 0,125 Кбайта содержит 128 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 4-символьный алфавит. Каждая страница содержит 40 строк по 132 символов в строке. Какой объем информации содержат 7 страниц текста?
5. Имеется файл с текстом из 14400 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 12 Кбайт. Поместится ли текст на диск?

Вариант 9

1. Сообщение, записанное буквами из 16-символьного алфавита, содержит 150 символов. Какой объем информации оно несет?
2. Сколько бит составляет сообщение, содержащее 0,5 Кбайта?
3. Информационное сообщение объемом 0,25 Кбайта содержит 256 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 32-символьный алфавит. Каждая страница содержит 42 строк по 105 символов в строке. Какой объем информации содержат 11 страниц текста?
5. Имеется файл с текстом из 18000 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 16 Кбайт. Поместится ли текст на диск?

Вариант 10

1. Сообщение, записанное буквами из 256-символьного алфавита, содержит 65 символов. Какой объем информации оно несет?
2. Сколько Кбайт составляет сообщение, содержащее 1,5 Мбайта?
3. Информационное сообщение объемом 3 Кбайта содержит 6144 символов. Сколько символов содержит алфавит, при помощи которого было записано это сообщение?
4. Для записи текста использовался 128-символьный алфавит. Каждая страница содержит 52 строк по 60символов в строке. Какой объем информации содержат 9 страниц текста?
5. Имеется файл с текстом из 24500 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 22 Кбайт. Поместится ли текст на диск?

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: §§3,4.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 6
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Представление чисел в компьютере.
Тип урока: Комбинированный урок.
Цель урока: изучить представление чисел в памяти компьютера
Образовательная: формирование знаний учащихся о формах представления числовой информации в компьютере; формирование практических навыков по представлению чисел в различных кодах.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
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 (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала
Показ видеоурока (6-1. Представление чисел в компьютере. Целые числа, 6-2. Представление чисел в компьютере. Вещественные числа)

Главные правила представления данных в компьютере

Если бы мы могли заглянуть в содержание компьютерной памяти, то увидели бы там примерно то, что условно изображено на рис. 1.5.

Рис. 1.5. Образ компьютерной памяти

Рисунок 1.5 отражает известное вам еще из курса информатики основной школы правило представления данных, которое назовем **правилом № 1**: данные (и программы) в памяти компьютера хранятся в двоичном виде, т. е. в виде цепочек единиц и нулей.

Современный компьютер может хранить и обрабатывать данные, представляющие информацию четырех видов: числовую, текстовую, графическую, звуковую. Двоичный код, изображенный на рис. 1.5, может относиться к любому виду данных.

Правило № 2: представление данных в компьютере дискретно.

Правило № 3: множество представимых в памяти компьютера величин ограничено и конечно.

Целые числа в компьютере

Правило № 4: в памяти компьютера числа хранятся в двоичной системе счисления*. С двоичной системой счисления вы знакомы из курса информатики 7-9 классов. Например, если под целое число выделяется ячейка памяти размером в 16 битов, то самое большое целое положительное число будет таким:

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

В десятичной системе счисления оно равно:

$2^{15} - 1 = 32\ 767$.

***** Конечно, и «внутри калькулятора» числа представляются в двоичном виде. Однако мы в это вдаваться не будем, рассмотрев лишь внешнее представление. Пример с калькулятором нам нужен был только для иллюстрации проблемы ограниченности.

Здесь первый бит играет роль знака числа. Ноль — признак положительного числа. Самое большое по модулю отрицательное число равно -32 768. Напомним (это было в курсе информатики основной школы), как получить его внутреннее представление:

1) перевести число 32 768 в двоичную систему счисления; это легко, поскольку $32\ 768 = 2^{15}$:
1000000000000000;

2) инвертировать этот двоичный код, т. е. заменить нули на единицы, а единицы — на нули:
0111111111111111;

3) прибавить единицу к этому двоичному числу (складывать надо по правилам двоичной арифметики), в результате получим:

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Единица в первом бите обозначает знак «минус». Не нужно думать, что полученный код — это «минус ноль». Этот код представляет число -32 768. Таковы правила машинного представления целых чисел. Данное представление называется **дополнительным кодом**.

Если под целое число в памяти компьютера отводится N битов, то диапазон значений целых чисел:
$[-2^{N-1}, 2^{N-1} - 1]$,

т. е. ограниченность целого числа в компьютере возникает из-за ограничений на размер ячейки памяти. Отсюда же следует и конечность множества целых чисел.

Мы рассмотрели **формат представления целых чисел со знаком**, т. е. положительных и отрицательных. Бывает, что нужно работать только с положительными целыми числами. В таком случае используется формат представления целых чисел без знака. В этом формате самое маленькое число — ноль (все биты — нули), а самое большое число для 16-разрядной ячейки:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

В десятичной системе это $2^{16} - 1 = 65\ 535$, примерно в два раза больше по модулю, чем в представлении со знаком.

Из всего сказанного делаем вывод: целые числа в памяти компьютера — это дискретное, ограниченное и конечное множество.

Границы множества целых чисел зависят от размера выделяемой ячейки памяти под целое число, а также от формата: со знаком или без знака. Шаг в компьютерном представлении последовательности целых чисел, как и в математическом, остается равным единице.

Рисунок 1.7 отражает то обстоятельство, что при переходе от математического представления множества целых чисел к представлению, используемому в информатике (компьютере), происходит переход к ограниченности и конечности.

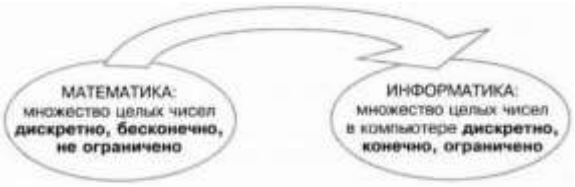


Рис. 1.7. Представление о множестве целых чисел в математике и в информатике

Вещественные числа в компьютере

Понятие вещественного (действительного) числа в математику ввел Исаак Ньютон в XVIII веке. В математике множество вещественных чисел непрерывно, бесконечно и не ограничено. Оно включает в себя множество целых чисел и еще бесконечное множество нецелых чисел. Между двумя любыми точками на числовой оси лежит бесконечное множество вещественных чисел, что и означает непрерывность множества.

Как мы говорили выше, числа в компьютере (в том числе и вещественные) представлены в двоичной системе счисления. Покажем, что множество вещественных чисел в компьютере дискретно, ограничено и конечно. Нетрудно догадаться, что это, так же как и в случае целых чисел, вытекает из ограничения размера ячейки памяти.

Снова для примера возьмем калькулятор с десятиразрядным индикаторным табло. Экспериментально докажем дискретность представления вещественных чисел. Выполним на калькуляторе деление 1 на 3. Из математики вам известно, что 1/3 — это рациональная дробь,

представление которой в виде десятичной дроби содержит бесконечное количество цифр: 0,3333333333... (3 в периоде). На табло калькулятора вы увидите:

	0.	3	3	3	3	3	3	3	3

Первый разряд зарезервирован под знак числа. После запятой сохраняется 8 цифр, а остальные не вмещаются в разрядную сетку (так это обычно называют). Значит, это не точное значение, равное 1/3, а его «урезанное» значение.

Следующее по величине число, которое помещается в разрядную сетку:

	0.	3	3	3	3	3	3	3	4

Оно больше предыдущего на 0,00000001. Это шаг числовой последовательности. Следовательно, два рассмотренных числа разделены между собой конечным отрезком. Очевидно, что предыдущее число такое:

	0.	3	3	3	3	3	3	3	2

Оно тоже отделено от своего «соседа справа» по числовой оси шагом 0,00000001. Отсюда делаем вывод: множество вещественных чисел, представимых в калькуляторе, дискретно, поскольку числа отделены друг от друга конечными отрезками.

А теперь выясним вот что: будет ли шаг в последовательности вещественных чисел на калькуляторе постоянной величиной (как у целых чисел)?

Вычислим выражение 100000/3. Получим:

	3	3	3	3	3.	3	3	3	3

Это число в 100 000 раз больше предыдущего и, очевидно, тоже приближенное. Легко понять, что следующее вещественное число, которое можно получить на табло калькулятора, будет больше данного на 0,0001. Шаг стал гораздо больше.

Отсюда приходим к выводу: множество вещественных чисел, представимых в калькуляторе, дискретно с переменной величиной шага между соседними числами.

Если отметить на числовой оси точные значения вещественных чисел, которые представимы в калькуляторе, то эти точки будут расположены вдоль оси неравномерно. Ближе к нулю — гуще, дальше от нуля — реже (рис. 1.8).

Рис. 1.8. Условное представление взаимного расположения множества вещественных чисел, представимых в компьютере

Все выводы, которые мы делаем на примере калькулятора, полностью переносятся на компьютер с переходом к двоичной системе счисления и с учетом размера ячейки компьютера, отводимой под вещественные числа. Неравномерное расположение вещественных чисел, представимых в компьютере, также имеет место.

Ответим на вопрос: ограничено ли множество вещественных чисел в памяти компьютера? Если продолжать эксперименты с калькулятором, то ответ на этот вопрос будет таким: да, множество вещественных чисел в калькуляторе ограничено.

Причиной тому служит все та же ограниченность разрядной сетки. Отсюда же следует и конечность множества.

Самое большое число у разных калькуляторов может оказаться разным. У самого простого это будет то же число, что мы получали раньше: 999999999. Если прибавить к нему единицу, то калькулятор выдаст сообщение об ошибке. А на другом, более «умном» и дорогом, калькуляторе прибавление единицы приведет к такому результату:

					1	e	+	0	9

Данную запись на табло надо понимать так: $1 \cdot 10^9$.

Такой формат записи числа называется форматом с плавающей запятой, в отличие от всех предыдущих примеров, где рассматривалось представление чисел в формате с фиксированной запятой.

Число, стоящее перед буквой «е», называется мантиссой, а стоящее после — порядком. «Умный калькулятор» перешел к представлению чисел в формате с плавающей запятой после того, как под формат с фиксированной запятой не стало хватать места на табло.

В компьютере то же самое: числа могут представляться как в формате с фиксированной запятой (обычно это целые числа), так и в формате с плавающей запятой.

Но и для форматы с плавающей запятой тоже есть максимальное число. В нашем «подопытном» калькуляторе это число:

	9	9	9	9	9	e	+	9	9

То есть 99999 • 10^{99}. Самое большое по модулю отрицательное значение -99999 • 10^{99}. Данные числа являются целыми, но именно они ограничивают представление любых чисел (целых и вещественных) в калькуляторе.

В компьютере все организовано аналогично, но предельные значения еще больше. Это зависит от разрядности ячейки памяти, выделяемой под число, и от того, сколько разрядов выделяется под порядок и под мантиссу.

Рассмотрим пример: пусть под все число в компьютере выделяется 8 байтов — 64 бита, из них под порядок — 2 байта, под мантиссу — 6 байтов. Тогда диапазон вещественных чисел, в переводе в десятичную систему счисления, оказывается следующим:

$\pm(5 \cdot 10^{-324} - 1{,}7 \cdot 10^{308})$.

Завершая тему, посмотрим на рис. 1.9. Смысл, заложенный в нем, такой: непрерывное, бесконечное и не ограниченное множество вещественных чисел, которое рассматривает математика, при его представлении в компьютере обращается в дискретное, конечное и ограниченное множество.

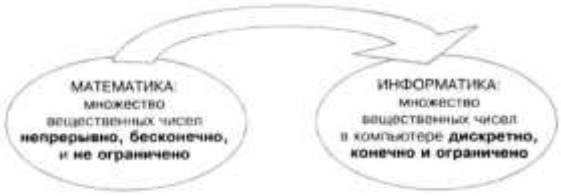


Рис. 1.9. Представление о множестве вещественных чисел в математике и в информатике

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

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

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 9
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №3 «Представление чисел».
Тип урока: Урок-практикум.
Цель урока: закрепление знаний о системах счисления и о представлении числе в памяти компьютера, полученных при изучении базового курса информатики.
Задачи урока:
образовательные: закрепление знаний о системах счисления и о представлении числе в памяти компьютера, полученных при изучении базового курса информатики;
развивающие: развивать культуру речи, мышление (умение сравнивать, анализировать, обобщать); учить ставить и разрешать проблемы, делать выводы;
воспитательная: воспитывать уважительное отношение к мнению других.
План урока:

1. Организационный момент (3 мин.)
2. Практическая работа (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Ход работы

Задание 1
Выписать алфавиты 2-ичной, 5-ричной, 8-ричной, 16-ричной систем счисления.

Задание 2
Записать первые 20 чисел натурального числового ряда в 2-ичной, 5-ричной, 8-ричной, 16-ричной системах счисления.

Задание 3
В каких системах счисления справедливо равенство:
а) 2 • 2 = 10; б) 2 • 3 = 11; в) 3 • 3 = 13?

Задание 4
Записать в развёрнутом виде числа.
A_{10} = 125,34; A_8 = 125,34; A_6 = 125,34; A_{16} = 125,34;
Пояснение. Развернутой формой записи числа называется запись вида:

$$A_q = \pm (a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + \dots + a_0q^0 + a_{-1}q^{-1} + a_{-1}q^{-1} + a_{-2}q^{-2} + \dots + a_{-m}q^{-m})$$

Здесь **Aq** - число, **q** - основание системы счисления, $\mathbf{a}_i$ - цифры данной системы счисления, **n** - количество разрядов целой части числа, **m** - количество разрядов дробной части числа.
Например:
$26{,}387_{10} = 2 \cdot 10^1 + 6 \cdot 10^0 + 3 \cdot 10^{-1} + 8 \cdot 10^{-2} + 7 \cdot 10^{-3}$;
$101{,}11_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$;
В последнем примере использована десятичная развёрнутая форма записи двоичного числа.

Задание 5
Перевести числа в десятичную систему счисления.

а) $A_8 = 341$; **б)** $A_6 = 341$; **в)** $A_{16} = 341$;
г) $A_5 = 34,1$; **д)** $A_{16} = E41A,12$

Задание 6

Перевести целые числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную системы:

а) 856; б) 664; в) 5012; г) 6435; д) 78.

Задание 7

Перевести десятичные дроби в двоичную и восьмеричную системы счисления, оставив пять знаков в дробной части нового числа.

а) 21,5; б) 432,54; в) 678,333.

Задание 8

Составить таблицы сложения и умножения в двоичной системе счисления и выполнить вычисления:

а) 1110 + 101; б) 10101 - 11; в) 101 • 11; г) 1110 / 10.

Задание 9

Представить числа в двоичном виде в восьмибитовой ячейке в формате

а) 5; б) 17; в) 64; г) 255.

Задание 10

Представить числа в двоичном виде в восьмибитовой ячейке в формате целого со знаком.

а) 56; б) -56; в) 127; г) -127.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: §5.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 10
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Представление текста, изображения и звука в компьютере.
Тип урока: Комбинированный урок.
Цель урока: изучить представление текста, изображения и звука в памяти компьютера
Образовательная: формирование знаний учащихся о формах представления числовой информации в компьютере; формирование практических навыков по представлению чисел в различных кодах.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
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 (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала
Показ видеоуроков с комментариями (7-1. Представление текста в компьютере, 7-2. Представление изображения в компьютере, 7-3. Представление звука в компьютере)

В этом параграфе обсудим способы компьютерного кодирования текстовой, графической и звуковой информации. С текстовой и графической информацией конструкторы «научили» работать ЭВМ, начиная с третьего поколения (1970-е годы). А работу со звуком «освоили» лишь машины четвертого поколения, современные персональные компьютеры. С этого момента началось распространение технологии мультимедиа.

Что принципиально нового появлялось в устройстве компьютеров с освоением ими новых видов информации? Главным образом, это периферийные устройства для ввода и вывода текстов, графики, видео, звука. Процессор же и оперативная память по своим функциям изменились мало. Существенно возросло их быстродействие, объем памяти. Но как это было на первых поколениях ЭВМ, так и осталось на современных ПК — основным навыком процессора в обработке данных является умение выполнять вычисления с двоичными числами. Обработка текста, графики и звука представляет собой тоже обработку числовых данных. Если сказать еще точнее, то это **обработка целых чисел**. По этой причине компьютерные технологии **называют цифровыми технологиями**.

О том, как текст, графика и звук сводятся к целым числам, будет рассказано дальше. Предварительно отметим, что здесь мы снова встретимся с **главной формулой информатики:**

$$2^i = N.$$

Смысл входящих в нее величин здесь следующий: **i** — разрядность ячейки памяти (в битах), **N** — количество различных целых положительных чисел, которые можно записать в эту ячейку.

Текстовая информация

Принципиально важно, что текстовая информация уже дискретна — состоит из отдельных знаков. Поэтому возникает лишь технический вопрос — как разместить ее в памяти компьютера.

Напомним о байтовом принципе организации памяти компьютеров, обсуждавшемся в курсе информатики основной школы. Каждая клеточка на нем обозначает бит памяти. Восемь подряд расположенных битов образуют байт памяти. Байты пронумерованы. Порядковый номер байта определяет его адрес в памяти компьютера. Именно по адресам процессор обращается к данным, читая или записывая их в память (рис. 1.10).

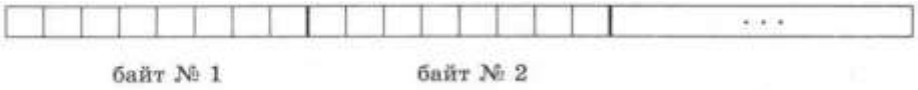


Рис. 1.10. Байтовая организация памяти

Модель представления текста в памяти весьма проста. За каждой буквой алфавита, цифрой, знаком препинания и иным общепринятым при записи текста символом закрепляется определенный двоичный код, длина которого фиксирована. В популярных системах кодировки (Windows-1251, KOI8 и др.) каждый символ заменяется на 8-разрядное целое положительное двоичное число; оно хранится в одном байте памяти. Это число является порядковым номером символа в кодовой таблице. Согласно главной формуле информатики, определяем, что размер алфавита, который можно закодировать, равен: 2^8 = 256. Этого количества вполне достаточно для размещения двух алфавитов естественных языков (английского и русского) и всех необходимых дополнительных символов.

Поскольку в мире много языков и много алфавитов, постепенно совершается переход на международную систему кодировки Unicode, в которой используются многобайтовые коды. Например, если код символа занимает 2 байта, то с его помощью можно закодировать 2^{16} = 65 536 различных символов.

При работе с электронной почтой почтовая программа иногда нас спрашивает, не хотим ли мы прибегнуть к кодировке Unicode для пересылаемых сообщений. Таким способом можно избежать проблемы несоответствия кодировок, из-за которой иногда не удается прочитать русский текст.

Текстовый документ, хранящийся в памяти компьютера, состоит не только из кодов символьного алфавита. В нем также содержатся коды, управляющие форматами текста при его отображении на мониторе или на печати: тип и размер шрифта, положение строк, поля и отступы и пр. Кроме того, текстовые процессоры (например, Microsoft Word) позволяют включать в документ и редактировать такие «нелинейные» объекты, как таблицы, оглавления, ссылки и гиперссылки, историю вносимых изменений и т. д. Всё это также представляется в виде последовательности байтовых кодов.

Графическая информация

Из курса информатики 7 - 9 классов вы знакомы с общими принципами компьютерной графики, с графическими технологиями. Здесь мы немного подробнее, чем это делалось раньше, рассмотрим способы представления графических изображений в памяти компьютера.

Принцип дискретности компьютерных данных справедлив и для графики. Здесь можно говорить о дискретном представлении изображения (рисунка, фотографии, видеокадров) и дискретности цвета.

Дискретное представление изображения

Изображение на экране монитора дискретно. Оно составляется из отдельных точек, которые называются пикселями (picture elements — элементы рисунка). Это связано с техническими особенностями устройства экрана, независимо от его физической реализации, будь то монитор на электронно-лучевой трубке, жидкокристаллический или плазменный. Эти «точки» столь близки друг другу, что глаз не различает промежутков между ними, поэтому изображение воспринимается как непрерывное, сплошное. Если выводимое из компьютера изображение формируется на бумаге (принтером или плоттером), то линии на нем также выглядят непрерывными. Однако в основе все равно лежит печать близких друг к другу точек.

В зависимости от того, на какое графическое разрешение экрана настроена операционная система компьютера, на экране могут размещаться изображения, имеющие размер 800 x 600, 1024 x 768 и более пикселей. Такая прямоугольная матрица пикселей на экране компьютера называется **растром**.

Качество изображения зависит не только от размера растра, но и от размера экрана монитора, который обычно характеризуется длиной диагонали. Существует параметр разрешения экрана. Этот параметр измеряется в точках на дюйм (по-английски dots per inch — dpi). У монитора с диагональю 15 дюймов размер изображения на экране составляет примерно 28 x 21 см. Зная, что в одном дюйме 25,4 мм, можно рассчитать, что при работе монитора в режиме 800 x 600 пикселей разрешение экранного изображения равно 72 dpi.

При печати на бумаге разрешение должно быть намного выше. Полиграфическая печать полноцветного изображения требует разрешения 200-300 dpi. Стандартный фотоснимок размером 10 x 15 см должен содержать примерно 1000 x 1500 пикселей.

Дискретное представление цвета

Восстановим ваши знания о кодировании цвета, полученные из курса информатики основной школы. Основное правило звучит так: любой цвет точки на экране компьютера получается путем

смешивания трех базовых цветов: красного, зеленого, синего. Этот принцип называется цветовой моделью RGB (Red, Green, Blue).

Двоичный код цвета определяет, в каком соотношении находятся интенсивности трех базовых цветов. Если все они смешиваются в одинаковых долях, то в итоге получается белый цвет. Если все три компоненты «выключены», то цвет пикселя — черный. Все остальные цвета лежат между белым и черным.

Дискретность цвета состоит в том, что интенсивности базовых цветов могут принимать конечное число дискретных значений.

Пусть, например, размер кода цвета пикселя равен 8 битам — 1 байту. Между базовыми цветами они могут быть распределены так:

К	К	З	З	З	С	С	С

2 бита — под красный цвет, 3 бита — под зеленый и 3 бита — под синий.

Интенсивность красного цвета может принимать $2^2 = 4$ значения, интенсивности зеленого и синего цветов — по $2^3 = 8$ значений. Полное число цветов, которые кодируются 8-разрядными кодами, равно: 4 - 8 - 8 = 256 = 2^8. Снова работает главная формула информатики.

Из описанного правила, в частности, следует:

Красный		Зеленый			Синий			
0	0	0	0	0	0	0	0	— код черного цвета
1	1	1	1	1	1	1	1	— код белого цвета
0	1	0	0	1	0	0	1	— код бледно-серого цвета
0	0	1	1	1	0	0	0	— код ярко-зеленого цвета
0	0	0	0	1	0	0	0	— код бледно-зеленого цвета

Обобщение этих частных примеров приводит к следующему правилу. Если размер кода цвета равен **b** битов, то количество цветов (размер палитры) вычисляется по формуле:

$K = 2^b$.

Величину **b** в компьютерной графике называют **битовой глубиной цвета**.

Еще один пример. Битовая глубина цвета равна 24. Размер палитры будет равен:

$K = 2^{24} = 16\,777\,216$.

В компьютерной графике используются разные цветовые модели для изображения на экране, получаемого путем излучения света, и изображения на бумаге, формируемого с помощью отражения света. Первую модель мы уже рассмотрели — это модель RGB. Вторая модель носит название CMYK.

Цвет, который мы видим на листе бумаги, — это отражение белого (солнечного) света. Нанесенная на бумагу краска поглощает часть палитры, составляющей белый цвет, а другую часть отражает. Таким образом, нужный цвет на бумаге получают путем «вычитания» из белого света «ненужных красок». Поэтому в цветной полиграфии действует не правило сложения цветов (как на экране компьютера), а правило вычитания. Мы не будем углубляться в механизм такого способа цветообразования.

Расшифруем лишь аббревиатуру CMYK: Cyan — голубой, Magenta — пурпурный, Yellow — желтый, black — черный.

Растровая и векторная графика

О двух технологиях компьютерной графики — растровой и векторной — вы знаете из курса информатики основной школы.

В растровой графике графическая информация — это совокупность данных о цвете каждого пикселя на экране. Это то, о чем говорилось выше. В векторной графике графическая информация — это данные, математически описывающие графические примитивы, составляющие рисунок: прямые, дуги, прямоугольники, овалы и пр. Положение и форма графических примитивов представляются в системе экранных координат.

Растровую графику (редакторы растрового типа) применяют при разработке электронных (мультимедийных) и полиграфических изданий. Растровые иллюстрации редко создают вручную с помощью компьютерных программ. Чаще для этой цели используют сканированные иллюстрации, подготовленные художником на бумаге, или фотографии. Для ввода растровых изображений в компьютер применяются цифровые фото- и видеокамеры. Большинство графических редакторов растрового типа в большей мере ориентированы не на создание изображений, а на их обработку.

Достоинство растровой графики — эффективное представление изображений фотографического качества. Основной недостаток растрового способа представления изображения — большой объем занимаемой памяти. Для его сокращения приходится применять различные способы сжатия данных. Другой недостаток растровых изображений связан с искажением изображения при его масштабировании. Поскольку изображение состоит из фиксированного числа точек, увеличение изображения приводит к тому, что эти точки становятся крупнее. Увеличение размера точек растра визуально искажает иллюстрацию и делает ее грубой.

Векторные графические редакторы предназначены в первую очередь для создания иллюстраций и в меньшей степени для их обработки.

Достоинства векторной графики — сравнительно небольшой объем памяти, занимаемой векторными файлами, масштабирование изображения без потери качества. Однако средствами векторной графики проблематично получить высококачественное художественное изображение. Обычно средства векторной графики используют не для создания художественных композиций, а для оформительских, чертежных и проектно-конструкторских работ.

Графическая информация сохраняется в файлах на диске. Существуют разнообразные форматы графических файлов. Они делятся на растровые и векторные. Растровые графические файлы (форматы JPEG, BMP, TIFF и другие) хранят информацию о цвете каждого пикселя изображения на экране. В графических файлах векторного формата (например, WMF, CGM) содержатся описания графических примитивов, составляющих рисунок.

Следует понимать, что графические данные, помещаемые в видеопамять и выводимые на экран, имеют растровый формат вне зависимости от того, с помощью каких программных средств (растровых или векторных) они получены.

Звуковая информация

Принципы дискретизации звука («оцифровки» звука) отражены на рис. 1.11.

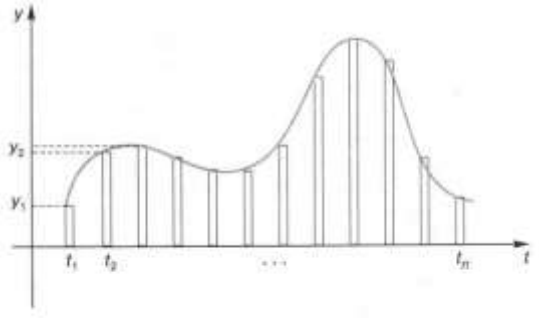


Рис. 1.11. Оцифровка звука
(y — интенсивность (уровень) звукового сигнала, t — время)

Ввод звука в компьютер производится с помощью звукового устройства (микрофона, радио и др.), выход которого подключается к порту **звуковой карты**. Задача звуковой карты — с определенной частотой производить измерения уровня звукового сигнала (преобразованного в электрические колебания) и результаты измерения записывать в память компьютера. Этот процесс называют оцифровкой звука.

Промежуток времени между двумя измерениями называется периодом измерений — τ с. Обратная величина называется **частотой дискретизации** — $1/\tau$ (герц). Чем выше частота измерений, тем выше качество цифрового звука.

Результаты таких измерений представляются целыми положительными числами с конечным количеством разрядов. Вы уже знаете, что в таком случае получается дискретное конечное множество значений в ограниченном диапазоне. Размер этого диапазона зависит от разрядности ячейки — регистра памяти звуковой карты. Снова работает формула 2^i, где i — разрядность регистра. Число i называют также разрядностью дискретизации. Записанные данные сохраняются в файлах специальных звуковых форматов.

Существуют программы обработки звука — редакторы звука, позволяющие создавать различные музыкальные эффекты, очищать звук от шумов, согласовывать с изображениями для создания мультимедийных продуктов и т. д. С помощью специальных устройств, генерирующих звук, звуковые файлы могут преобразовываться в звуковые волны, воспринимаемые слухом человека.

При хранении оцифрованного звука приходится решать проблему уменьшения объема звуковых файлов. Для этого кроме кодирования данных без потерь, позволяющего осуществлять стопроцентное восстановление данных из сжатого потока, используется кодирование данных с потерями. Цель такого кодирования — добиться схожести звучания восстановленного сигнала с оригиналом при максимальном сжатии данных. Это достигается путем использования различных алгоритмов, сжимающих оригинальный сигнал путем выкидывания из него слабослышимых элементов. Методов сжатия, а также программ, реализующих эти методы, существует много.

Для сохранения звука без потерь используется универсальный звуковой формат файлов WAV. Наиболее известный формат «сжатого» звука (с потерями) — MP3. Он обеспечивает сжатие данных в 10 раз и более.

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: §6, вопросы к параграфу устно.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 11
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №4 «Представление текстов. Сжатие текстов».
Тип урока: Урок-практикум.
Цель урока: практическое закрепление знаний о представлении в компьютере текстовых данных.
Задачи урока:
образовательные: практическое закрепление знаний о представлении в компьютере текстовых данных;
развивающие: развивать культуру речи, мышление (умение сравнивать, анализировать, обобщать); учить ставить и разрешать проблемы, делать выводы;
воспитательная: воспитывать уважительное отношение к мнению других.
План урока:

1. Организационный момент (3 мин.)
2. Практическая работа (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Ход работы

Задание 1
Закодировать текст ***Happy Birthday to you!!*** с помощью кодировочной таблицы ASCII

Стандартная часть таблицы ASCII

№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код
32	пробел	00100000	56	8	00111000	80	P	01010000	104	h	01101000
33	!	00100001	57	9	00111001	81	Q	01010001	105	i	01101001
34	"	00100010	58	:	00111010	82	R	01010010	106	j	01101010
35	#	00100011	59	;	00111011	83	S	01010011	107	k	01101011
36	$	00100100	60	<	00111100	84	T	01010100	108	l	01101100
37	%	00100101	61		00111101	85	U	01010101	109	m	01101101
38	&	00100110	62	>	00111110	86	V	01010110	110	n	01101110
39	'	00100111	63	?	00111111	87	W	01010111	111	o	01101111
40	(	00101000	64	@	01000000	88	X	01011000	112	p	01110000
41	)	00101001	65	A	01000001	89	Y	01011001	113	q	01110001
42	*	00101010	66	B	01000010	90	Z	01011010	114	r	01110010
43	+	00101011	67	C	01000011	91	[	01011011	115	s	01110011
44	,	00101100	68	D	01000100	92	\	01011100	116	t	01110100
45	-	00101101	69	E	01000101	93	]	01011101	117	u	01110101
46	.	00101110	70	F	01000110	94	^	01011110	118	v	01110110
47	/	00101111	71	G	01000111	95	_	01011111	119	w	01110111
48	0	00110000	72	H	01001000	96	`	01100000	120	x	01111000
49	1	00110001	73	I	01001001	97	a	01100001	121	y	01111001
50	2	00110010	74	J	01001010	98	b	01100010	122	z	01111010
51	3	00110011	75	K	01001011	99	c	01100011	123	{	01111011
52	4	00110100	76	L	01001100	100	d	01100100	124	|	01111100
53	5	00110101	77	M	01001101	101	e	01100101	125	}	01111101
54	6	00110110	78	N	01001110	102	f	01100110	126	~	01111110
55	7	00110111	79	O	01001111	103	g	01100111	127		01111111

Записать двоичное и шестнадцатеричное представление кода (для записи шестнадцатеричного кода использовать средство для просмотра файлов любого файлового менеджера).

Задание 2

Декодировать текст, записанный в международной кодировочной таблице ASCII (дано десятичное представление).

Стандартная часть таблицы ASCII

№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код
32	пробел	00100000	56	8	00111000	80	P	01010000	104	h	01101000
33	!	00100001	57	9	00111001	81	Q	01010001	105	i	01101001
34	"	00100010	58	:	00111010	82	R	01010010	106	j	01101010
35	#	00100011	59	;	00111011	83	S	01010011	107	k	01101011
36	$	00100100	60	<	00111100	84	T	01010100	108	l	01101100
37	%	00100101	61		00111101	85	U	01010101	109	m	01101101
38	&	00100110	62	>	00111110	86	V	01010110	110	n	01101110
39	'	00100111	63	?	00111111	87	W	01010111	111	o	01101111
40	(	00101000	64	@	01000000	88	X	01011000	112	p	01110000
41	)	00101001	65	A	01000001	89	Y	01011001	113	q	01110001
42	*	00101010	66	B	01000010	90	Z	01011010	114	r	01110010
43	+	00101011	67	C	01000011	91	[	01011011	115	s	01110011
44	,	00101100	68	D	01000100	92	\	01011100	116	t	01110100
45	-	00101101	69	E	01000101	93	]	01011101	117	u	01110101
46	.	00101110	70	F	01000110	94	^	01011110	118	v	01110110
47	/	00101111	71	G	01000111	95	_	01011111	119	w	01110111
48	0	00110000	72	H	01001000	96	`	01100000	120	x	01111000
49	1	00110001	73	I	01001001	97	a	01100001	121	y	01111001
50	2	00110010	74	J	01001010	98	b	01100010	122	z	01111010
51	3	00110011	75	K	01001011	99	c	01100011	123	{	01111011
52	4	00110100	76	L	01001100	100	d	01100100	124	|	01111100
53	5	00110101	77	M	01001101	101	e	01100101	125	}	01111101
54	6	00110110	78	N	01001110	102	f	01100110	126	~	01111110
55	7	00110111	79	O	01001111	103	g	01100111	127		01111111

72 101 108 108 111 44 32 109 121 32 102 114 105 101 110 100 33

Задание 3

Пользуясь таблицей кодировки ASCII, расшифровать текст, представленный в виде двоичных кодов символов.

Стандартная часть таблицы ASCII

№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код
32	пробел	00100000	56	8	00111000	80	P	01010000	104	h	01101000
33	!	00100001	57	9	00111001	81	Q	01010001	105	i	01101001
34	"	00100010	58	:	00111010	82	R	01010010	106	j	01101010
35	#	00100011	59	;	00111011	83	S	01010011	107	k	01101011
36	$	00100100	60	<	00111100	84	T	01010100	108	l	01101100
37	%	00100101	61		00111101	85	U	01010101	109	m	01101101
38	&	00100110	62	>	00111110	86	V	01010110	110	n	01101110
39	'	00100111	63	?	00111111	87	W	01010111	111	o	01101111
40	(	00101000	64	@	01000000	88	X	01011000	112	p	01110000
41	)	00101001	65	A	01000001	89	Y	01011001	113	q	01110001
42	*	00101010	66	B	01000010	90	Z	01011010	114	r	01110010
43	+	00101011	67	C	01000011	91	[	01011011	115	s	01110011
44	,	00101100	68	D	01000100	92	\	01011100	116	t	01110100
45	-	00101101	69	E	01000101	93	]	01011101	117	u	01110101
46	.	00101110	70	F	01000110	94	^	01011110	118	v	01110110
47	/	00101111	71	G	01000111	95	_	01011111	119	w	01110111
48	0	00110000	72	H	01001000	96	`	01100000	120	x	01111000
49	1	00110001	73	I	01001001	97	a	01100001	121	y	01111001
50	2	00110010	74	J	01001010	98	b	01100010	122	z	01111010
51	3	00110011	75	K	01001011	99	c	01100011	123	{	01111011
52	4	00110100	76	L	01001100	100	d	01100100	124	|	01111100
53	5	00110101	77	M	01001101	101	e	01100101	125	}	01111101
54	6	00110110	78	N	01001110	102	f	01100110	126	~	01111110
55	7	00110111	79	O	01001111	103	g	01100111	127		01111111

01010000 01100101 01110010 01101101 00100000 01010101
01101110 01101001 01110110 01100101 01110010 01110011
01101001 01110100 01111001

Задание 4

Пользуясь кодовой страницей Windows-1251 таблицы кодировки ASCII, получить шестнадцатеричный код слова ИНФОРМАТИЗАЦИЯ.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
90	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A0	NBSP 00A0	Ў 040E	ў 045E	Ј 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B0	° 00B0	± 00B1	І 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	ј 0458	Ѕ 0405	ѕ 0455	ї 0457
C0	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D0	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E0	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F0	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: §6.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 12
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №5 «Представление изображения и звука».
Тип урока: Урок-практикум.
Цель урока: практическое закрепление знаний о представлении в компьютере графических данных и звука.
Задачи урока:
образовательные: практическое закрепление знаний о представлении в компьютере графических данных и звука;
развивающие: развивать культуру речи, мышление (умение сравнивать, анализировать, обобщать); учить ставить и разрешать проблемы, делать выводы;
воспитательная: воспитывать уважительное отношение к мнению других.
План урока:

1. Организационный момент (3 мин.)
2. Практическая работа (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Ход работы

Справочная информация

В некоторых заданиях используется модельный (учебный) вариант монитора с размером растра 10x10 пикселей.

При векторном подходе изображение рассматривается как совокупность простых элементов: прямых линий, дуг, окружностей, эллипсов, прямоугольников, закрасок и пр., которые называются графическими примитивами. Графическая информация — это данные, однозначно определяющие все графические примитивы, составляющие рисунок.

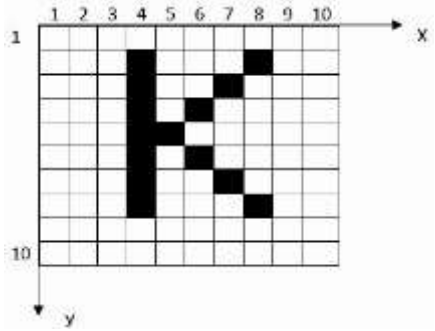
Положение и форма графических примитивов задаются в системе графических координату связанных с экраном. Обычно начало координат расположено в верхнем левом углу экрана. Сетка пикселей совпадает с координатной сеткой. Горизонтальная ось X направлена слева направо; вертикальная ось У — сверху вниз.

Отрезок прямой линии однозначно определяется указанием координат его концов; окружность — координатами центра и радиусом; многоугольник — координатами его углов, закрашенная область — граничной линией и цветом закраски и пр.

Учебная система векторных команд представлена в таблице.

Установить X, Y	Установить текущую позицию (X, Y)
Линия к X1, Y1	Нарисовать линию от текущей позиции в позицию (X1, Y1), позиция X1, Y1 становится текущей
Линия X1, Y1, X2, Y2	Нарисовать линию с координатами начала X1, Y1 и координатами конца X2, Y2. Текущая позиция не устанавливается
Окружность X, Y, R	Нарисовать окружность; X, Y – координаты центра, R – длина в пикселях
Эллипс X1, Y1, X2, Y2	Нарисовать эллипс, ограниченный прямоугольником; (X1, Y1)-координаты левого верхнего, а (X2, Y2) – правого нижнего угла этого прямоугольника
Прямоугольник X1, Y1, X2, Y2	Нарисовать прямоугольник; (X1, Y1)-координаты левого верхнего, а (X2, Y2) – правого нижнего угла этого прямоугольника
Цвет_рисования ЦВЕТ	Установить текущий цвет рисования
Цвет_закраски ЦВЕТ	Установить текущий цвет закраски
Закрасить X, Y, ЦВЕТ ГРАНИЦЫ	Закрасить произвольную замкнутую фигуру; X, Y – координаты любой точки внутри замкнутой фигуры, ЦВЕТ ГРАНИЦЫ – цвет граничной линии

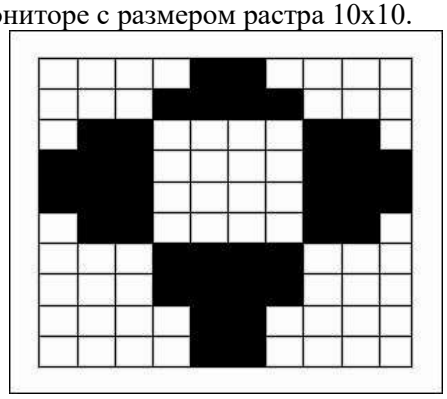
Например, требуется написать последовательность получения изображения буквы К:



Изображение буквы «К» на рисунке описывается тремя векторными командами:
Линия(4, 2, 4, 8)
Линия(5, 5, 8, 2)
Линия(5, 5, 8, 8)

Задание 1

Построить двоичный код приведенного черно-белого растрового изображения, полученного на мониторе с размером растра 10x10.



Задание 2

Определить, какой объем памяти требуется для хранения 1 бита изображения на вашем компьютере (для этого нужно через **Свойства экрана** определить битовую глубину цвета).

Задание 3

Битовая глубина цвета равна 24. Сколько различных оттенков серого цвета может быть отображено на экране (серый цвет получается, если уровни яркости всех трех базовых цветов одинаковы)?

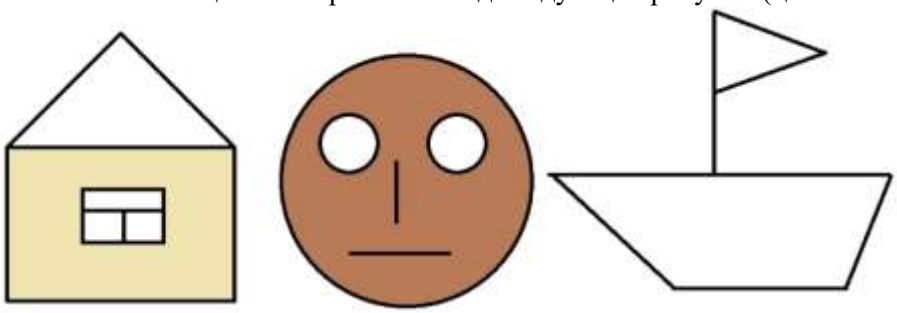
Задание 4

Дан двоичный код 8-цветного изображения. Размер монитора — 10x10 пикселей. Что изображено на рисунке (зарисовать)?

001 111 111 111 010 010 111 111 111 001
111 111 111 011 011 011 011 111 111 111
111 111 011 111 111 111 111 011 111 111
111 011 111 111 111 111 111 111 011 111
110 011 111 111 110 110 111 111 011 110
110 011 111 111 110 110 111 111 011 110
111 011 111 111 111 111 111 111 011 111
111 111 011 111 111 111 111 011 111 111
111 111 111 011 011 011 011 111 111 111
001 111 111 111 010 010 111 111 111 001

Задание 5

Описать с помощью векторных команд следующие рисунки (цвет заливки произвольный).



Задание 6

Получить растровое и векторное представления всех цифр от 0 до 9.

Задание 7

По приведенному ниже набору векторных команд определить, что изображено на рисунке (зарисовать).

Цвет рисования Голубой
Прямоугольник 12, 2, 18, 8
Прямоугольник 10, 1, 20, 21
Прямоугольник 20, 6, 50, 21
Цвет рисования Желтый
Цвет закраски Зеленый
Окружность 20, 24, 3
Окружность 40, 24, 3
Закрасить 20, 24, Желтый
Закрасить 40, 24, Желтый
Цвет закраски Голубой
Закрасить 30, 10, Голубой
Закрасить 15, 15, Голубой
Цвет закраски Розовый
Закрасить 16, 6, Голубой

3. Подведение итогов урока

Оценки за урок следующие _____

4. Домашнее задание

Записываем домашнее задание: §6.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 13
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Хранение и передача информации.
Тип урока: Урок-лекция.
Цели урока:
Образовательная: формирование знаний учащихся о способах и носителях хранения информации
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
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 (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала
Показ видеоуроков с комментариями (8. Сбор, хранение и передача информации)

Из курса основной школы вам известно:
Человек хранит информацию в собственной памяти, а также в виде записей на различных внешних (по отношению к человеку) носителях: на камне, папирусе, бумаге, магнитных и оптических носителях и пр. Благодаря таким записям, информация передается не только в пространстве (от человека к человеку), но и во времени — из поколения в поколение.

Рассмотрим способы хранения информации более подробно.
Информация может храниться в различных видах: в виде записанных текстов, рисунков, схем, чертежей; фотографий, звукозаписей, кино- или видеозаписей. В каждом случае применяются свои носители.
Носитель — это материальная среда, используемая для записи и хранения информации.
Практически носителем информации может быть любой материальный объект. Информацию можно сохранять на камне, дереве, стекле, ткани, песке, теле человека и т. д. Здесь мы не станем обсуждать различные исторические и экзотические варианты носителей. Ограничимся современными средствами хранения информации, имеющими массовое применение.

Использование бумажных носителей информации
Носителем, имеющим наиболее массовое употребление, до сих пор остается бумага. Изобретенная во II веке н. э. в Китае бумага служит людям уже 19 столетий.
Для сопоставления объемов информации на разных носителях будем пользоваться единицей — байтом, считая, что один знак текста «весит» 1 байт. Нетрудно подсчитать информационный объем книги, содержащей 300 страниц с размером текста на странице примерно 2000 символов. Текст такой книги имеет объем примерно 600 000 байтов, или 586 Кб. Средняя школьная библиотека, фонд которой составляют 5000 томов, имеет информационный объем приблизительно 2861 Мб = 2,8 Гб.

Что касается долговечности хранения документов, книг и прочей бумажной продукции, то она очень сильно зависит от качества бумаги, красителей, используемых при записи текста, условий хранения.

Интересно, что до середины XIX века (с этого времени для производства бумаги начали использовать древесину) бумага делалась из хлопка и текстильных отходов — тряпья. Чернилами служили натуральные красители. Качество рукописных документов того времени было довольно высоким, и они могли храниться тысячи лет. С переходом на древесную основу, с распространением машинописи и средств копирования, с началом использования синтетических красителей срок хранения печатных документов снизился до 200-300 лет.

На первых компьютерах бумажные носители использовались для цифрового представления вводимых данных. Это были перфокарты: картонные карточки с отверстиями, хранящие двоичный код вводимой информации. На некоторых типах ЭВМ для тех же целей применялась перфорированная бумажная лента.

Использование магнитных носителей информации

В XIX веке была изобретена магнитная запись. Первоначально она использовалась только для сохранения звука. Самым первым носителем магнитной записи была стальная проволока диаметром до 1 мм. В начале XX столетия для этих целей использовалась также стальная катаная лента. Тогда же (в 1906 г.) был выдан и первый патент на магнитный диск. Качественные характеристики всех этих носителей были весьма низкими. Достаточно сказать, что для производства 14-часовой магнитной записи устных докладов на Международном конгрессе в Копенгагене в 1908 г. потребовалось 2500 км, или около 100 кг проволоки.

В 20-х годах XX века появляется магнитная лента сначала на бумажной, а позднее — на синтетической (лавсановой) основе, на поверхность которой наносится тонкий слой ферромагнитного порошка. Во второй половине XX века на магнитную ленту научились записывать изображение, появляются видеокамеры, видеомагнитофоны.

На ЭВМ первого и второго поколений магнитная лента использовалась как единственный вид сменного носителя для устройств внешней памяти. Любая компьютерная информация на любом носителе хранится в двоичном (цифровом) виде. Поэтому независимо от вида информации: текст это, или изображение, или звук — ее объем можно измерить в битах и байтах. На одну катушку с магнитной лентой, использовавшейся в лентопротяжных устройствах первых ЭВМ, помещалось приблизительно 500 Кб информации.

С начала 1960-х годов в употребление входят компьютерные магнитные диски: алюминиевые или пластмассовые диски, покрытые тонким магнитным порошковым слоем толщиной в несколько микрон. Информация на диске располагается по круговым концентрическим дорожкам, на которые она записывается и считывается в процессе вращения диска с помощью магнитных головок.

На первых ПК использовались гибкие магнитные диски (флоппи-диски) — сменные носители информации с небольшим объемом памяти — до 2 Мб. Начиная с 1980-х годов, в ПК начали использоваться встроенные в системный блок накопители на жестких магнитных дисках, или НЖМД (англ. HDD — Hard Disk Drive). Их еще называют винчестерскими дисками.

Винчестерский диск представляет собой пакет магнитных дисков, надетых на общую ось, которая при работе компьютера находится в постоянном вращении. С каждой магнитной поверхностью пакета дисков контактирует своя магнитная головка.

Информационная емкость современных винчестерских дисков **измеряется в терабайтах**.

Оптические диски и флеш-память

Применение оптического, или лазерного, способа записи информации начинается в 1980-х годах. Его появление связано с изобретением квантового генератора — лазера, источника очень тонкого (толщина порядка микрона) луча высокой энергии. Луч способен выжигать на поверхности

плавкого материала двоичный код данных с очень высокой плотностью. Считывание происходит в результате отражения от такой «перфорированной» поверхности лазерного луча с меньшей энергией («холодного» луча). Первоначально на ПК вошли в употребление оптические компакт - диски — CD, информационная емкость которых составляет от 190 Мб до 700 Мб.

Во второй половине 1990-х годов появились цифровые универсальные видеодиски DVD (Digital Versatile Disk) с большой емкостью, измеряемой в гигабайтах (до 17 Гб). Увеличение их емкости по сравнению с CD связано с использованием лазерного луча меньшего диаметра, а также двухслойной и двусторонней записи. Вспомните пример со школьной библиотекой. Весь ее книжный фонд можно разместить на одном DVD.

В настоящее время оптические диски (CD и DVD) являются наиболее надежными материальными носителями информации, записанной цифровым способом. Эти типы носителей бывают как однократно записываемыми — пригодными только для чтения, так и перезаписываемыми — пригодными для чтения и записи.

В последнее время появилось множество мобильных цифровых устройств: цифровые фото- и видеокамеры, MP3-плееры, карманные компьютеры, мобильные телефоны, устройства для чтения электронных книг, GPS-навигаторы и др. Все эти устройства нуждаются в переносных носителях информации. Но поскольку все мобильные устройства довольно миниатюрные, к носителям информации для них предъявляются особые требования. Они должны быть компактными, обладать низким энергопотреблением при работе, быть энергонезависимыми при хранении, иметь большую емкость, высокие скорости записи и чтения, долгий срок службы. Всем этим требованиям удовлетворяют флеш-карты памяти. Информационный объем флеш-карты может составлять несколько десятков гигабайтов.

В качестве внешнего носителя для компьютера широкое распространение получили так называемые **флеш-брелоки** (их называют в просторечии «флешки»), выпуск которых начался в 2001 году. Большой объем информации, компактность, высокая скорость чтения/записи, удобство в использовании — основные достоинства этих устройств.

Флеш-брелок подключается к USB-порту компьютера и позволяет скачивать данные со скоростью около 10 Мб в секунду.

В последние годы активно ведутся работы по созданию еще более **компактных носителей информации с использованием нанотехнологий**, работающих на уровне атомов и молекул вещества. В результате один компакт-диск, изготовленный по нанотехнологии, сможет заменить тысячи оптических дисков. По предположениям экспертов, приблизительно через 20 лет плотность хранения информации возрастет до такой степени, что на носителе объемом примерно с кубический сантиметр можно будет записать каждую секунду человеческой жизни.

Из курса основной школы вам известно:

- **Распространение информации происходит в процессе ее передачи.**
- **Процесс передачи информации протекает от источника к приемнику по информационным каналам связи.**

Ранее уже говорилось о том, что первой в истории технической системой передачи информации стал **телеграф.** В 1876 году американец Александр Белл изобрел телефон. На основании открытия немецким физиком Генрихом Герцем электромагнитных волн (1886 год), А. С. Попов в России в 1895 году и почти одновременно с ним в 1896 году Г. Маркони в Италии изобрели радио. Телевидение и Интернет появились в XX веке.

Модель передачи информации К. Шеннона

Все перечисленные способы информационной связи основаны на передаче на расстояние физического (электрического или электромагнитного) сигнала и подчиняются некоторым общим законам. Исследованием этих законов занимается теория связи, возникшая в 1920-х годах. **Математический аппарат теории связи — математическую теорию связи разработал американский ученый Клод Шеннон**. Клодом Шенноном была предложена модель процесса передачи информации по техническим каналам связи, представленная схемой на рис. 2.1.

Рис. 2.1. Модель передачи информации по техническим каналам связи

Работу такой схемы можно пояснить на знакомом всем процессе разговора по телефону. Источником информации является говорящий человек. Кодирующим устройством — микрофон телефонной трубки, с помощью которого звуковые волны (речь) преобразуются в электрические сигналы. Каналом связи служит телефонная сеть (провода, коммутаторы телефонных узлов, через которые проходит сигнал). Декодирующим устройством является телефонная трубка (наушник) слушающего человека — приемника информации. Здесь пришедший электрический сигнал превращается в звук.

В теме "Информация. Представление информации" уже говорилось о кодировании на примере передачи информации через письменный документ. Кодирование там было определено как процесс представления информации в виде, удобном для ее хранения и/или передачи.

Применительно к процессу передачи информации по технической системе связи ***под кодированием понимается любое преобразование информации, идущей от источника, в форму, пригодную для ее передачи по каналу связи***.

Современные компьютерные системы передачи информации — компьютерные сети, работают по тому же принципу. Есть процесс кодирования, преобразующий двоичный компьютерный код в физический сигнал того типа, который передается по каналу связи. Декодирование заключается в обратном преобразовании передаваемого сигнала в компьютерный код. Например, при использовании телефонных линий в компьютерных сетях функции кодирования/декодирования выполняет прибор, который называется модемом.

Пропускная способность канала и скорость передачи информации

Разработчикам технических систем передачи информации приходится решать две взаимосвязанные задачи: как обеспечить наибольшую скорость передачи информации и как уменьшить потери информации при передаче. К. Шеннон был первым ученым, взявшимся за решение этих задач и создавшим новую для того времени науку — **теорию информации**.

Шеннон определил способ измерения количества информации, передаваемой по каналам связи. Им было введено понятие **пропускной способности канала** *как максимально возможной скорости передачи информации*. Эта скорость измеряется в битах в секунду (а также килобитах в секунду, мегабитах в секунду).

Пропускная способность канала связи зависит от его технической реализации. Например, в компьютерных сетях используются следующие **средства связи:**

- **телефонные линии;**
- **электрическая кабельная связь;**
- **оптоволоконная кабельная связь;**
- **радиосвязь**.

Пропускная способность телефонных линий — десятки и сотни Кбит/с; *пропускная способность оптоволоконных линий и линий радиосвязи* измеряется десятками и сотнями Мбит/с.

Скорость передачи информации связана не только с пропускной способностью канала связи. Представьте себе, что текст на русском языке, содержащий 1000 знаков, передается с использованием двоичного кодирования. В первом случае используется телеграфная 5-разрядная кодировка. Во втором случае — компьютерная 8-разрядная кодировка. Тогда длина кода сообщения в первом случае составит 5000 битов, во втором случае — 8000 битов. При передаче по одному и тому же каналу второе сообщение будет передаваться дольше в 1,6 раза (8000/5000). Отсюда, казалось бы, следует вывод: длину кода сообщения нужно делать минимально возможной.

Однако существует другая проблема, которая на рис. 2.1 отмечена словом «шум».

Шум, защита от шума

Термином «шум» называют разного рода помехи, искажающие передаваемый сигнал и приводящие к потере информации. Такие помехи, прежде всего, возникают по техническим причинам, таким как плохое качество линий связи, незащищенность друг от друга различных потоков информации, передаваемых по одним и тем же каналам. Существуют и другие источники помех, имеющие физическое происхождение.

Иногда, например, беседуя по телефону, мы слышим шум, треск, мешающие понять собеседника, или на наш разговор накладывается разговор других людей.

Наличие шума приводит к потере передаваемой информации. В таких случаях необходима защита от шума. Для этого в первую очередь применяются технические способы защиты каналов связи от воздействия шумов. Такие способы бывают самыми разными, иногда простыми, иногда очень сложными. Например: использование экранированного кабеля вместо «голого» провода; применение разного рода фильтров, отделяющих полезный сигнал от шума и пр.

Шеннон разработал специальную **теорию кодирования**, дающую методы борьбы с шумом. Одна из важных идей этой теории состоит в том, что передаваемый по линии связи код должен быть избыточным. За счет этого потеря какой-то части информации при передаче может быть компенсирована. Например, если при разговоре по телефону вас плохо слышно, то, повторяя каждое слово дважды, вы имеете больше шансов на то, что собеседник поймет вас правильно.

Владимир Александрович Котельников (1908–2005), Россия

В системах передачи информации используется так называемое **помехоустойчивое кодирование**, вносящее определенную избыточность.

Однако нельзя делать избыточность слишком большой. Это приведет к задержкам и удорожанию связи. Теория кодирования как раз и позволяет получить такой код, который будет оптимальным: избыточность передаваемой информации будет минимально возможной, а достоверность принятой информации — максимальной.

Большой вклад в научную теорию связи внес известный советский ученый Владимир Александрович Котельников. В 1940-1950-х годах им получены фундаментальные научные результаты по проблеме помехоустойчивости систем передачи информации.

В современных системах цифровой связи для борьбы с потерей информации при передаче часто применяется следующий прием.

Все сообщение разбивается на порции — блоки. Для каждого блока вычисляется контрольная сумма (сумма двоичных цифр), которая передается вместе с данным блоком.

В месте приема заново вычисляется контрольная сумма принятого блока и, если она не совпадает с первоначальной суммой, передача данного блока повторяется. Так происходит до тех пор, пока исходная и конечная контрольные суммы не совпадут.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§7,8, задания №8,9 (§7) и задания №7,8 (§8) письменно в тетради. Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 14
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №6 «Управление алгоритмическим исполнителем».
Тип урока: Урок-практикум.
Цели урока: закрепление навыков программного управления учебными исполнителями алгоритмов.
Задачи урока:
Образовательная: закрепление навыков программного управления учебными исполнителями алгоритмов.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (3 мин.)
2. Практическая работа (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

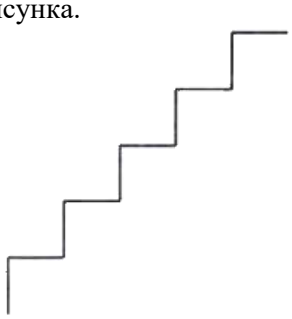
1. Организационный момент
Здравствуйте ребята, садитесь. Проверка готовности к уроку.

2. Практическая работа

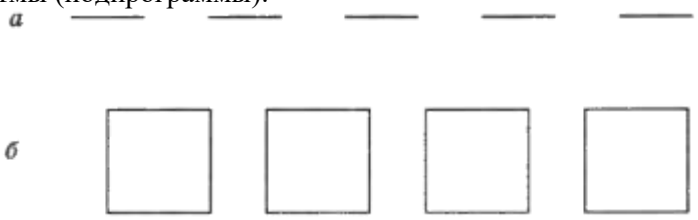
Ход работы

Используемое программное обеспечение: *среда какого-либо учебного исполнителя алгоритмов графического типа, назначение которого — рисование на экране компьютера. К числу таких исполнителей относятся: Черепашка Лого, Чертежник, Кенгуренок и др.*

Задание 1
Написать подпрограмму (процедуру) STEP и с ее помощью составить программу рисования лесенки по диагонали через все поле рисунка.

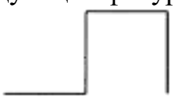


Задание 2
Написать программы для рисования следующих рисунков на всю ширину поля, используя вспомогательные алгоритмы (подпрограммы).



Задание 3

Описать подпрограмму для рисования следующей фигуры.

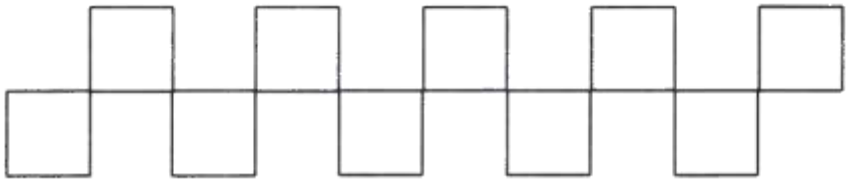


Задание 4

Используя подпрограмму из предыдущего задания, составить программу для рисования «забора» через все поле рисунка.

Задание 5

Оформить решение задания 4 в виде подпрограммы и с ее помощью составить программу рисования следующей фигуры.



3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §9, вопросы к параграфу устно.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 15
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Автоматическая обработка информации.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся о технических системах обработки информации и создания алгоритмов.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
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 (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Проверка домашнего задания

3. Объяснение нового материала
Показ видеоуроков с комментариями (12. Автоматическая обработка информации)


Пост Эмиль Леон
(1897 — 1954)
США

В качестве примера автомата, выполняющего обработку информации, рассмотрим машину Э. Поста. Алгоритм, по которому работает машина Поста, будем называть программой.

Договоримся о терминологии: под словом «программа» мы всегда будем понимать алгоритм, записанный по строгим правилам языка команд исполнителя — на языке программирования для данного исполнителя.

Рис. 2.3. Модель машины Поста

Опишем архитектуру машины Поста (рис. 2.3).

Имеется бесконечная информационная лента, разделенная на позиции — клетки. В каждой клетке может либо стоять метка (некоторый знак), либо отсутствовать (пусто).

Вдоль ленты движется каретка — считывающее устройство. На рисунке 2.3 она обозначена стрелкой.

Каретка может передвигаться шагами: один шаг — смещение на одну клетку вправо или влево. Клетку, под которой установлена каретка, будем называть текущей.

Каретка является еще и процессором машины.

С ее помощью машина может:

- распознать, пустая клетка или помеченная знаком;
- стереть знак в текущей клетке;
- записать знак в пустую текущую клетку.

Если произвести замену меток на единицы, а пустых клеток — на нули, то информацию на ленте можно будет рассматривать как аналог двоичного кода телеграфного сообщения или данных в памяти компьютера. Существенное отличие каретки - процессора машины Поста от процессора компьютера состоит в том, что **в компьютере возможен доступ процессора к ячейкам памяти в произвольном порядке**, а в **машине Поста — только последовательно**.

Назначение машины Поста — производить преобразования на информационной ленте. Исходное состояние ленты можно рассматривать как исходные данные задачи, конечное состояние ленты — как результат решения задачи. Кроме того, в исходные данные входит информация о начальном положении каретки.

Теперь рассмотрим систему команд машины Поста (табл. 2.1). Запись всякой команды начинается с ее порядкового номера в программе — **n**. Затем следует код операции и после него — номер следующей выполняемой команды программы — **m**.

Рассмотрим пример программы решения задачи на машине Поста. Исходное состояние показано на рис. 2.3. Машина должна стереть знак в текущей клетке и присоединить его слева к группе знаков, расположенных справа от каретки. Программа приведена в табл. 2.2.

Таблица 2.1. **Система команд машины Поста**

Команда	Действие
$n \leftarrow m$	Сдвиг каретки на шаг влево и переход к выполнению команды с номером m
$n \rightarrow m$	Сдвиг каретки на шаг вправо и переход к выполнению команды с номером m
$n \vee m$	Запись метки в текущую пустую клетку и переход к выполнению команды с номером m
$n \mid m$	Стирание метки в текущей клетке и переход к выполнению команды с номером m
n !	Остановка выполнения программы
n ? m,k	Переход в зависимости от содержимого текущей клетки: если текущая клетка пустая, то следующей будет выполняться команда с номером m, если непустая — команда с номером k

Таблица 2.2. **Программа для машины Поста**

Команда	Действие
1 | 2	Стирание метки; переход к следующей команде
2 → 3	Сдвиг вправо на один шаг
3 ? 2,4	Если клетка пустая, то переход к команде 2, иначе — к команде 4
4 ← 5	Сдвиг влево на шаг (команда выполнится, когда каретка выйдет на первый знак группы)
5 ∨ 6	Запись метки в пустую клетку
6 !	Остановка машины

В процессе выполнения приведенной программы многократно повторяется выполнение команд с номерами 2 и 3. Такая ситуация называется ***циклом***. Напомним, что цикл относится к числу основных алгоритмических структур вместе ***со следованием и ветвлением***.

А теперь научим машину Поста играть в интеллектуальную игру, которая называется «Игра Баше».

Опишем правила игры.

Играют двое. Перед ними 21 (или 16, или 11 и т. д.) фишка. Игроки берут фишки по очереди. За один ход можно взять от 1 до 4 фишек. Проигрывает тот, кто забирает последнюю фишку.

Имеется выигрышная тактика для игрока, берущего фишки вторым. Она заключается в том, чтобы брать такое количество фишек, которое дополняет число фишек, взятых соперником на предыдущем ходе, до пяти.

Роль фишек на информационной ленте машины Поста будут выполнять метки (знаки). Машина играет с человеком. Человеку предоставляется возможность стирать метки (брать фишки) первым. Машина будет вступать в игру второй.

Исходная обстановка: на ленте массив из 21 клетки содержит метки. Каретка установлена на крайней слева клетке этого массива. Стирать метки можно только подряд. Выигрышным результатом должна быть одна оставшаяся метка перед очередным ходом человека.

Еще раз напомним принцип выигрышной тактики: стирать столько меток, чтобы в сумме с метками, стертыми противником за предыдущий ход, их было пять.

Программа управления машиной Поста в игре Баше против человека приведена в табл. 2.3.

Таблица 2.3. **Программа игры Баше**

Команда	Действие
1 ? 2,1	Машина ждет появления пустой клетки над кареткой. После очередного хода человека машина делает свой ход. Если человек видит всего одну метку на ленте, он прекращает игру, признав свое поражение
2 → 3	Эта серия команд выведет каретку на пятую (десятую, пятнадцатую и т. д.) позицию. Какой бы ход ни сделал соперник, в ней обязательно будет стоять метка
3 → 4	
4 → 5	
5 → 6	
6 ↓ 7	Стирается метка в текущей клетке
7 ← 8	Шаг влево
8 ? 9,6	Если клетка не пустая, то возврат к команде 6
9 → 10	Каретка перемещается к первой помеченной клетке. После этого машина возвращается к команде 1 и ждет хода человека (или признания им своего поражения)
10 ? 9,1	

Действуя по данной программе и начиная стирать метки второй после человека, машина всегда будет выигрывать, если правильно задано начальное число меток, которое должно быть равно 5n + 1, где **n** — любое натуральное число. В противном случае машина может проиграть.

4. Подведение итогов урока

Оценки за урок, следующие _____

5. Домашнее задание

Записываем домашнее задание: §10, вопросы к параграфу устно.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 16
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №7 «Автоматическая обработка информации».
Тип урока: Урок-практикум.
Цели урока: знакомство с основами теории алгоритмов на примере решения задач на программное управление алгоритмической машиной Поста.
Задачи урока:
Образовательная: закрепление навыков программного управления учебными исполнителями алгоритмов.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:
1. Организационный момент (3 мин.)
2. Практическая работа (32 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

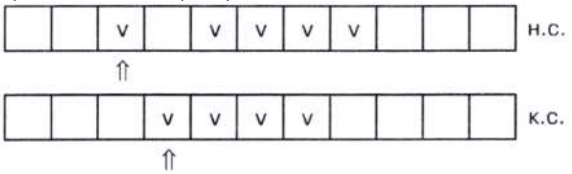
2. Практическая работа

Ход работы

Используемое программное обеспечение: *имитатор машины Поста.*
Система команд машины Поста: (везде буква ***n*** *обозначает номер текущей команды):*

Команда	Действие
$n \leftarrow m$	Сдвиг каретки на шаг влево и переход к выполнению команды с номером m
$n \rightarrow m$	Сдвиг каретки на шаг вправо и переход к выполнению команды с номером m
$n \vee m$	Установа метки в текущую пустую клетку
$n \updownarrow m$	Стирание метки в текущей клетке
n !	Остановка выполнения программы
n ? m, k	Переход по содержимому текущей клетки: если текущая клетка пустая, то следующей будет выполняться команда с номером m; если в текущей клетке стоит метка, то выполнится команда с номером k

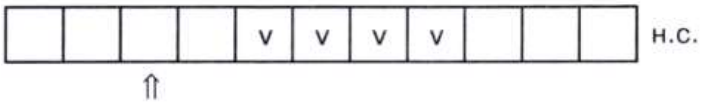
Задание 1
Составить программу перевода информационной ленты машины Поста из начального состояния (н.с.) в конечное (к.е.):



Задание 2

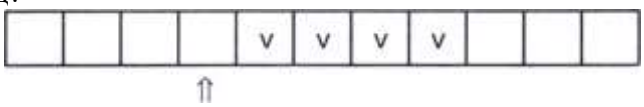
1. Выполнить на машине Поста программу:

1 v 2	4 ← 5
2 → 3	5 v 6
3 ? 2, 4	6 !



2. Какую задачу решает исполнитель по этой программе?

3. Что произойдет, если начальное состояние информационной ленты будет иметь следующий вид?



В следующих задачах считается, что ***n*** *расположенных подряд меток обозначают число* ***n*** *(непозиционная система счисления с основанием 1).*

Задание 3

Написать для машины Поста программу сложения двух чисел, записанных на ленте и расположенных через одну пустую клетку друг от друга. Начальное положение каретки — под пустой клеткой, отделяющей числа.

Задание 4

Написать для машины Поста программу вычитания двух чисел, разделенных одной пустой клеткой. Уменьшаемое не меньше вычитаемого. Начальное положение каретки — под пустой клеткой, отделяющей уменьшаемое от вычитаемого.

Указание. Стирать метки по одной у каждого числа, пока у вычитаемого не кончатся все метки.

Задание 5

Используя программу вычитания, проверить, что получится, если:

а) уменьшаемое равно вычитаемому;

б) уменьшаемое меньше вычитаемого.

Задание 6

Написать для машины Поста программу деления числа, записанного метками, на 2. Исходное число должно делиться на 2 без остатка.

Указание. Стереть каждую вторую метку; уплотнить оставшиеся метки.

Задание 7

Используя программу деления числа на 2: а) проверить, что получится для числа 2; б) модифицировать программу с учетом числа 2.

Указание. Справа от пустой клетки поставить метку, а слева стереть две метки. Так поступать до тех пор, пока слева остаются метки.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §10.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 17
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Информационные процессы в компьютере.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся об информационных процессах в компьютере.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
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 (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала
Показ видеоуроков с комментариями (12. Автоматическая обработка информации)

Из курса основной школы вам известно:

• Компьютер (ЭВМ) — автоматическое, программно-управляемое устройство для работы с информацией.

• В состав компьютера входят устройства памяти (хранение данных и программ), процессор (обработка информации), устройства ввода/вывода (прием/передача информации).

• В 1946 году Джоном фон Нейманом были сформулированы основные принципы устройства ЭВМ, которые называют фон-неймановской архитектурой.

• Современный компьютер представляет собой единство аппаратуры (hardware) и программного обеспечения (software).

Серийное производство электронных вычислительных машин (ЭВМ) начинается в разных странах в 1950-х годах. Историю развития ЭВМ принято делить на поколения. Переход от одного поколения к другому связан со сменой элементной базы, на которой создавались машины, с изменением архитектуры ЭВМ, с развитием основных технических характеристик (скорости вычислений, объема памяти и др.), с изменением областей применения и способов эксплуатации машин.

Под архитектурой ЭВМ понимаются наиболее общие принципы построения компьютера, реализующие программное управление его работой и взаимодействие основных функциональных узлов.

В основе архитектуры ЭВМ разных поколений лежат принципы Джона фон Неймана. Однако в процессе развития происходят некоторые отклонения от фон-неймановской архитектуры.

Однопроцессорная архитектура ЭВМ

Элементной базой ЭВМ **первого поколения** (1950-годы) были электронные лампы, а ЭВМ **второго поколения** (1960-е годы) создавались на базе полупроводниковых элементов. Однако их архитектура была схожей. Она в наибольшей степени соответствовала принципам фон Неймана. В этих машинах один процессор управлял работой всех устройств: внутренней

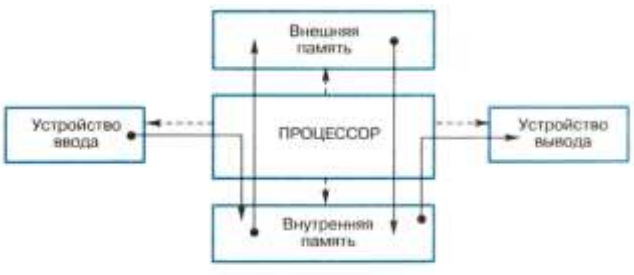


Рис. 2.4. Структура однопроцессорной ЭВМ. Сплошные стрелки — передача данных, пунктирные стрелки — управляющее воздействие

и внешней памяти, устройств ввода и вывода, как показано на рис. 2.4.

Согласно принципам фон Неймана, исполняемая программа хранится во внутренней памяти — в оперативном запоминающем устройстве (ОЗУ). Там же находятся данные, с которыми работает программа. Каждая команда программы и каждая величина (элемент данных) занимают определенные ячейки памяти, как показано на рис. 2.5.

	Внутренняя память	
	Номер ячейки	Содержимое ячейки
Программа	1	
	2	
	. . .	
	. . .	
	N	Команда STOP
Данные	$N+1$	Величина 1
	$N+2$	Величина 2
	. . .	. . .

Рис. 2.5. Размещение в ОЗУ программы и данных

Процессор начинает выполнение программы с первой команды и заканчивает на команде остановки, назовем ее STOP. При выполнении очередной команды процессор извлекает из памяти обрабатываемые величины и заносит их в специальные ячейки внутренней памяти процессора — регистры. Затем выполняется команда, например складываются два числа, после чего полученный результат записывается в определенную ячейку памяти. Процессор переходит к выполнению следующей команды. Исполнение программы закончится, когда процессор обратится к команде STOP.

Среди команд программы существуют команды обработки данных и команды обращения к внешним устройствам. Команды обработки данных выполняет сам процессор с помощью входящего в него арифметико-логического устройства — АЛУ, и этот процесс происходит сравнительно быстро. А команды управления внешними устройствами выполняются самими этими устройствами: устройствами ввода/вывода, внешней памятью. Время выполнения этих команд во много раз больше, чем время выполнения команд обработки данных. При однопроцессорной архитектуре ЭВМ, показанной на рис. 2.4, процессор, отдав команду внешнему устройству, ожидает завершения ее выполнения. При большом числе обращений к внешним устройствам может оказаться, что большую часть времени выполнения программы процессор «простаивает» и, следовательно, его КПД оказывается низким. Быстродействие ЭВМ с такой архитектурой находилось в пределах 10-20 тысяч операций в секунду (оп./с).

Использование периферийных процессоров

Следующим шагом в развитии архитектуры ЭВМ стал отказ от однопроцессорного устройства. Уже на последних моделях машин второго поколения, помимо центрального процессора (ЦП), выполнявшего обработку данных, присутствовали периферийные процессоры, которые назывались каналами ввода/вывода (рис. 2.6). Их задача состояла в автономном управлении устройствами ввода/вывода и внешней памяти, что освобождало от этой работы центральный процессор. В результате КПД центрального процессора существенно возрос. Быстродействие некоторых моделей машин с такой архитектурой составляло от 1 до 3 млн оп./с.

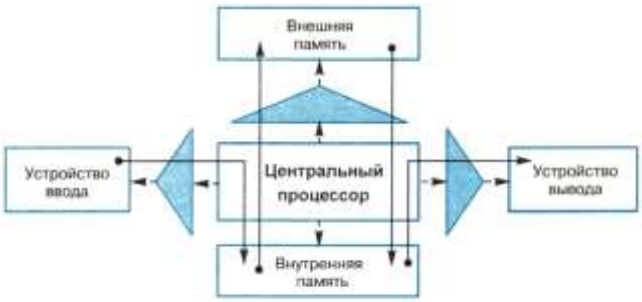


Рис. 2.6. Структура ЭВМ с одним центральным процессором и периферийными процессорами управления внешними устройствами (треугольники)

На всех моделях ЭВМ третьего поколения, которые создавались на базе интегральных схем (1970-80-е годы), использовалась архитектура с одним центральным процессором и периферийными процессорами внешних устройств. Такая многопроцессорная архитектура позволяла реализовать мультипрограммный режим работы: пока одна программа занята вводом/выводом данных, которым управляет периферийный процессор, другая программа занимает центральный процессор, выполняя вычисления. Благодаря совершенствованию элементной базы и других аппаратных средств на некоторых моделях ЭВМ третьего поколения достигалось быстродействие до 10 млн оп./с.

Для разделения ресурсов ЭВМ между несколькими выполняемыми программами потребовалось создание специального программного обеспечения: операционной системы (ОС). К разделяемым ресурсам, прежде всего, относятся время работы центрального процессора и оперативная память. Задача ОС состоит в том, чтобы разные программы, выполняемые одновременно на ЭВМ, «не мешали» друг другу и чтобы КПД центрального процессора был максимальным, иначе говоря, чтобы ЦП не «простаивал». ОС берет на себя также заботу об очередности использования несколькими программами общих внешних устройств: внешней памяти, устройств ввода/вывода.

Архитектура персонального компьютера

Персональный компьютер (ПК) — самый распространенный в наше время тип компьютера. Появление ПК связано с созданием микропроцессоров, которое началось в 1970-х годах. До недавнего времени в устройстве ПК существовал один центральный процессор и множество периферийных процессоров, управляющих внешними устройствами, которые называются контроллерами. Архитектура такого ПК изображена на рис. 2.7.

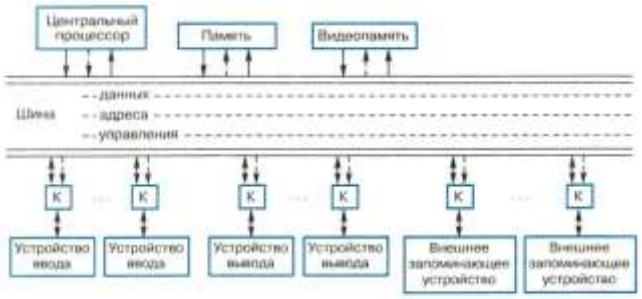


Рис. 2.7. Архитектура персонального компьютера (сплошные стрелки — направление потоков информации, пунктирные — направление управляющих сигналов, К — контроллер)

Для связи между отдельными функциональными узлами ПК используется общая информационная магистраль, которая называется системной шиной.

Системная шина состоит из трех частей:

• шина данных (для передачи данных);

• шина адреса (для передачи адресов устройств, которым передаются данные);

• шина управления (для передачи управляющих сигналов, синхронизирующих работу разных устройств).

Важное достоинство такой архитектуры возможность подключения к компьютеру новых устройств или замена старых устройств на более современные. Это называется принципом открытой архитектуры. Для каждого типа и модели устройства используется свой контроллер, а в составе операционной системы имеется управляющая программа, которая называется драйвером устройства.

Открытая архитектура персонального компьютера — это архитектура, предусматривающая модульное построение компьютера с возможностью добавления и замены отдельных устройств».

Важное событие в совершенствовании архитектуры ПК произошло в 2005 году: был создан первый двухъядерный микропроцессор. Каждое ядро способно выполнять функции центрального процессора. Эта особенность архитектуры позволяет производить на ПК параллельную обработку данных, что существенно увеличивает его производительность. Выпускаемые в настоящее время микропроцессоры содержат до 8 ядер.

Архитектура ненеймановских вычислительных систем

Несмотря на стремительно нарастающую производительность ЭВМ, которая каждые 4 5 лет по важнейшим показателям практически удваивается, всегда есть классы задач, для которых никакой производительности не хватает. Укажем некоторые из них.

1. Математические расчеты, лежащие в основе реализации математических моделей многих процессов. Гигантские вычислительные ресурсы, которые можно реализовать очень быстро (как иногда говорят, в реальном масштабе времени), необходимы для более надежного и долгосрочного прогноза погоды, для решения аэрокосмических задач, в том числе и оборонных, для решения многих инженерных задач и т. д.

2. Поиск информации в гигантских базах данных, в информационном пространстве Интернета.

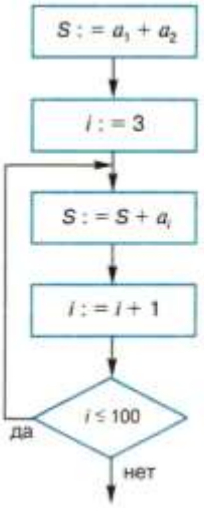
3. Моделирование интеллекта — при всех фантастических показателях, объем оперативной памяти современных компьютеров составляет лишь малую долю объема памяти человека.

Быстродействие компьютера с одним центральным процессором имеет физическое ограничение: повышение тактовой частоты процессора ведет к повышению тепловыделения, которое не может быть неограниченным. Перспективный путь повышения производительности компьютера лежит на пути отказа от единственности главных устройств компьютера: либо процессора, либо оперативной памяти, либо шины, либо всего этого вместе. Это путь еще большего отступления от архитектуры фон Неймана.

Чтобы стало понятнее, зачем компьютеру несколько процессоров, обсудим алгоритм решения простейшей математической задачи. Есть массив из 100 чисел: a_1, a_2, ... , a_{100}. Требуется найти их сумму.

Нет ничего проще! И на компьютере, и без него мы, скорее всего, поступим так: сложим первые два числа, как-то обозначим их сумму (например, S), затем прибавим к ней третье, и будем делать это еще 98 раз. Это пример последовательного вычислительного процесса. Его блок-схема приведена на рис. 2.8.

Поскольку у человека нет второй головы, иначе эту задачу в одиночку не решить. Но представим, что мы решаем ее не в одиночку, а всем классом (25 человек). Тогда возникает возможность совсем иной последовательности действий.



1. Объединим числа в пары — по два на каждого (итого распределили 50 чисел); например, ученик № 1 берет себе a_1 и a_2, ученик № 2 — a_3 и a_4, и т. д.

2. Даем команду «складывай!» — и каждый складывает свои числа.

3. Даем команду «записывай!» — и каждый записывает мелом на классной доске свой результат.

4. Поскольку у нас осталось еще 50 необработанных чисел (a_{51}, ..., a_{100}), повторяем пункты 1 - 3. После этого имеем на доске 50 чисел $b_1 = a_1 + a_2$, ..., $b_{50} = a_{99} + a_{100}$ — результаты парных сложений.

5. Объединим в пары числа b_i и повторим выполнение пунктов 2 - 4.

Продолжаем этот процесс (2 - 5) до тех пор, пока не останется одно число — искомая сумма.

Первое впечатление, что очень сложно, гораздо сложнее, чем алгоритм на рис. 2.8. Если бы мы захотели записать этот алгоритм в виде блок-схемы, то нам бы пришлось кроме описания порядка и объектов действий сделать то, что мы никогда при записи алгоритмов не делали, — предусмотреть синхронизацию параллельных процессов по времени. Например, выполнение команд 2 и 3 должно завершиться всеми участниками вычислений до того, как они будут продолжены (до перехода к п. 4), иначе даже при решении этой простой задачи наступит хаос.

Но сложность не есть объективная причина отвергнуть такой путь, особенно если речь идет о возможности значительного ускорения компьютерных вычислений. То, что мы предложили выше, называется на языке программистов распараллеливанием вычислений и вполне поддается формальному описанию. Эффект ускорения вычислений очевиден: пункт 2 в приведенном выше алгоритме ускоряет соответствующий этап работы в 25 раз!

Следующий вопрос: что надо изменить в устройстве компьютера, чтобы он смог так работать? Для реализации подобной схемы вычислений компьютеру потребуется 25 процессоров, объединенных в одну архитектуру и способных работать параллельно.

Такие **многопроцессорные вычислительные комплексы** — реальность сегодняшней вычислительной техники.

Вернемся, однако, к описанной выше последовательности действий — в ней еще есть источники проблем. Представим себе, что в схеме на рис. 2.7 мы дорисовали еще 24 центральных процессора, соединенных с шиной. При реализации в таком компьютере команды 3 произойдет одновременное обращение 25 процессоров к системной шине для пересылки результатов сложения в оперативную память. Но поскольку шина одна, числа по ней могут пересылаться только по одному! Значит, для выполнения команды 3 придется организовать очередь на передачу чисел в память. Тут же возникает вопрос: не сведет ли к нулю эта очередь все преимущества от параллельности выполнения операций на шаге 2? А если преимущества останутся, то насколько они велики? Окупятся ли расходы на 24 дополнительных процессора?

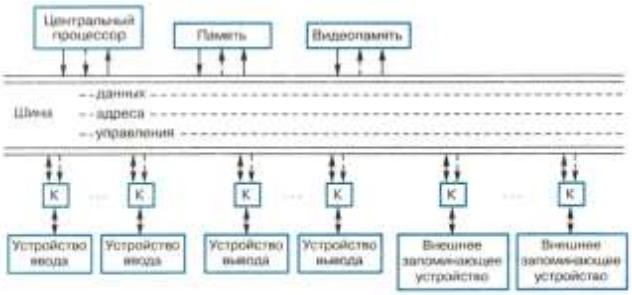


Рис. 2.7. Архитектура персонального компьютера (сплошные стрелки — направление потоков информации, пунктирные — направление управляющих сигналов, К — контроллер)

В возникшей ситуации естественен следующий шаг «изобретательской мысли»: ввод в архитектуру нескольких системных шин. А если еще подумать над возможными проблемами, то и нескольких устройств оперативной памяти.

Как видите, все это очень непросто! Обсуждаемые изменения в устройстве компьютера приводят к «ненеймановским» архитектурам. Изобретателям таких систем приходится искать компромисс между возрастающей сложностью (и, как следствие, — стоимостью) и ускорением их работы.

Варианты реализации ненеймановских вычислительных систем

В самом общем смысле под параллельными вычислениями понимаются процессы обработки данных, в которых одновременно могут выполняться нескольких машинных операций. Параллельные вычисления реализуются как за счет новой архитектуры вычислительной техники, так и за счет новых технологий программирования. Такие технологии называются **параллельным программированием**.

Распределенные вычисления — способ реализации параллельных вычислений путем использования множества компьютеров, объединенных в сеть. Такие вычислительные системы еще называют **мультикомпьютерными**.

Распределенные вычисления часто реализуются с помощью **компьютерных кластеров** — нескольких компьютеров, связанных в локальную сеть и объединенных специальным программным обеспечением, реализующим параллельный вычислительный процесс. Распределенные вычисления могут производиться и с помощью **многомашинных вычислительных комплексов**, образуемых объединением нескольких отдельных компьютеров через глобальные сети.

Мультипроцессорные системы образуют единый компьютер, который относится к классу суперкомпьютеров. Достижение параллелизма в них происходит благодаря возможности независимой работы отдельных устройств и их дублирования: несколько процессоров, блоков оперативной памяти, шин и т. д. Мультипроцессорная система может использовать разные способы доступа к общей для всей системы памяти. Если все процессоры имеют равный (однородный) доступ к единой памяти, то соответствующая вычислительная система называется **векторным суперкомпьютером**.

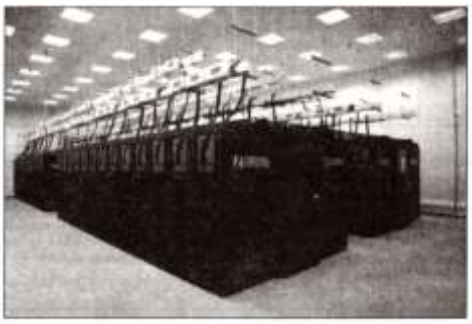


Рис. 2.9. Суперкомпьютер «Ломоносов» (МГУ им. М. В. Ломоносова)

Один из самых мощных в мире суперкомпьютеров под названием «Ломоносов» (рис. 2.9) произведен в России и работает в Московском государственном университете. Его быстродействие составляет более ста триллионов операций в секунду.

4. Подведение итогов урока

Оценки за урок, следующие _____

5. Домашнее задание

Записываем домашнее задание: §11, вопросы к параграфу устно.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 18
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №8 «Проектное задание Выбор конфигурации компьютера».
Тип урока: Урок-практикум.
Цели урока:
Образовательная:

- знакомство с основными техническими характеристиками устройств персонального компьютера;
- знакомство с номенклатурой и символикой;
- знакомство с принципами комплектации при покупке ПК;
- получение навыков в оценке стоимости комплекта устройств ПК.

Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Проверка домашнего задания (4 мин.)
3. Итоговая проверочная работа (30 мин.)
4. Подведение итогов урока (3 мин.)
5. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

3. Итоговая проверочная работа

Справочная информация

При сборке компьютера из отдельных комплектующих необходимо учитывать два основных момента. Первый из них касается круга задач, для решения которых будет использоваться компьютер. Условно компьютеры можно разделить на несколько групп в зависимости от их функционального предназначения: офисные, учебные, игровые, домашние, мультимедийные и т. д. Предназначение компьютера определяет тот набор устройств, из которых он должен состоять, а также их основные характеристики. Например, для офисного компьютера необходимо наличие принтера, а игровому не обойтись без мощного процессора, большого объема ОП, качественной видеокарты с достаточным объемом видеопамяти и хорошего монитора.

Второй момент касается совместимости отдельных устройств с материнской платой. Прежде всего, это относится к совместимости по интерфейсу подключения.

Интерфейс — это стандарт присоединения компонентов к системе.

В качестве интерфейсов служат разъемы на материнской плате.

Существует несколько различных процессорных интерфейсов, для каждого из которых выпускаются свои модели материнских плат. Для процессоров фирмы Intel, например, в 2012 году использовались интерфейсы Socket 775, Socket 1155, Socket 1156, Socket 1366, а для процессоров фирмы AMD — Socket FM1, Socket AM2, Socket AM3. Поэтому при выборе материнской платы всегда в первую очередь следует обращать внимание на ее процессорный интерфейс.

Для видеокарт в настоящее время используется два интерфейса подключения: AGP 8x (устаревший) и PCI-Express x16 (обычно его обозначают PCI-E).

Оперативная память обычно имеет тип DDR (устаревший), DDR-II или DDR-III и соответствующие интерфейсы подключения к материнской плате. Иногда на одной материнской плате могут одновременно присутствовать два типа разъемов.

Современные жесткие диски подключаются к интерфейсам SerialATA-II или SerialATA-III (обозначаются SATA-II и SATA-III). Существуют также переносные жесткие диски, подключаемые к интерфейсу USB.

При комплектации компьютера необходимо учитывать, что некоторые компоненты могут быть встроены непосредственно в материнскую плату (видеокарты, звуковые карты, сетевые карты) и приобретение дополнительных аналогичных устройств может быть оправдано только в том случае, если они имеют лучшие

характеристики, чем интегрированные. Обычно для обозначения встроенной звуковой карты используется название кодека AC'97, а для встроенной сетевой карты — обозначение LAN, после которого указывается пропускная способность в Мбитах в секунду. Встроенные видеокарты могут обозначаться либо их названием, либо просто сокращением «в/к».

Пример 1
Мат.плата Soc-775 ASRock G31M-S rev2.0 (RTL)PCI-E+SVGA+LAN SATA MicroATX 2DDR-II
Материнская плата с Socket 775. Производитель — ASRock. Есть встроенная видеокарта и сетевая карта (SVGA и LAN). Имеется интерфейс подключения PCI-Express (для внешней видеокарты). Имеются интерфейс подключения жестких дисков SerialATA. Имеется два разъема для оперативной памяти типа DDR-II с максимальной пропускной способностью 6400 Мбайт/с.

Пример 2
Процессор Soc-AM3 AMD ATHLON II X2 240 (2.80 ГГц/2Мб/4000МГц) OEM(ADX240O)
Двухъядерный (X2) процессор фирмы AMD Athlon-II с сокетом AM-3. Тактовая частота — 2,8 Ггц, реальная тактовая частота — 2200 МГц.

Пример 3
В/к PCI-E 512Mb DDR RadeonHD4650 Palit (RTL) 128bit, DVI, HDMI
Видеокарта с интерфейсом PCI-Express xl6. Тип видеопамяти — DDR, объем видеопамяти — 512 Мбайт.

Задание 1

В компьютерном салоне имеется следующий набор устройств:

- процессор Socket-1155 Intel Celeron, 2,5 ГГц — 1980 руб.;
- процессор Socket-1155 Intel Core i3-2100, 3,1 ГГц — 4390 руб.;
- процессор Socket-1155 Intel Core i5-3450, 3,1 ГГц — 6740 руб.;
- процессор Socket-AM3 AMD ATHLON II X3, 3,1 ГГц — 2510 руб.;
- процессор Socket-AM3 AMD Phenom II X4, 3,5 ГГц — 5510 руб.;
- материнская плата Socket-1155 ASRock DDR3 mATX AC'97+LAN+ VGA — 1850 руб.;
- материнская плата Socket-1155 ASUSTeK 2xPCI-E+GbLAN SATA 2DDR-III — 2760 руб.;
- материнская плата Socket-1155 GigaByte 2xPCI-E+GbLAN SATA — 4180 руб.;
- материнская плата Socket-775 ASUSTeK PCI-E+SVGA+ GbLAN SATA 4DDR-III — 1770 руб.;
- материнская плата Socket-AM3 ASUSTeK PCI-E+GbLAN SATA 4DDR-III — 3300 руб.;
- корпус компьютера с блоком питания мощностью 350 В — 1310 руб.;
- корпус компьютера с блоком питания мощностью 400 В — 2480 руб.;
- модули оперативной памяти объемом 1 Гбайт — 640 руб.;
- модули оперативной памяти объемом 2 Гбайта — 850 руб.;
- модули оперативной памяти объемом 4 Гбайта — 1330 руб.;
- модули оперативной памяти объемом 8 Гбайт — 2010 руб.;
- жесткий диск объемом 500 Гбайт — 3100 руб.;
- жесткий диск объемом 1 Тбайт — 3570 руб.;
- жесткий диск объемом 2 Тбайта — 4800 руб.;
- видеокарта с объемом видеопамяти 512 Мбайт — 1270 руб.;
- видеокарта с объемом видеопамяти 1 Гбайт — 2700 руб.;
- видеокарта с объемом видеопамяти 2 Гбайта — 5880 руб.;
- звуковая карта — 960 руб.;
- звуковая карта (профессиональная) — 5770 руб.;
- звуковые колонки 5 В — 730 руб.;
- звуковые колонки 5.1 (5 колонок+сабвуфер) — 2920 руб.;
- сетевая карта 10/100/1000 Мбит/с — 570 руб.;
- привод CD-RW — 830 руб.;
- привод CD-RW/DYD-RW — 970 руб.;
- принтер струйный (цветной) — 2150 руб.;
- фотопринтер струйный — 2500 руб.;
- принтер лазерный — 4630 руб.;
- сканер — 2510 руб.;
- модем — 940 руб.;
- монитор LCD, диагональ 19 дюймов — 3600 руб.;
- монитор LCD, диагональ 20 дюймов — 5870 руб.;
- монитор LCD, диагональ 20 дюймов — 8540 руб.;

• мышь оптическая — 200 руб.;
• клавиатура — 180 руб.;
• клавиатура мультимедийная — 1250 руб.

Подобрать комплектующие для компьютера, предназначенного для решения определенного круга задач (см. варианты ниже). При выборе компонент компьютера необходимо уложиться в заданную сумму. Для подбора различных вариантов решения указанной задачи использовать табличный процессор (электронные таблицы).

Вариант 1.

Офисный компьютер, предназначенный в основном для работы с текстовыми документами и выхода в Интернет через локальную сеть организации. Сумма — 22 000 руб.

Вариант 2.

Домашний компьютер, предназначенный в основном для компьютерных игр, просмотра видеофильмов и выхода в Интернет через телефонную линию связи. Сумма — 30 000 руб.

Вариант 3.

Мультимедийный компьютер, предназначенный для выполнения видеомонтажа и создания рекламных видеороликов. Сумма — 35 000 руб. (обязательно два жестких диска).

Вариант 4.

Учебный компьютер, предназначенный для обучения школьников информатике с выходом в локальную сеть учебного заведения. Сумма — 20 000 руб.

Вариант 5.

Домашний компьютер, предназначенный для работы с документами, обработки фотографий, создания фонограмм и выхода в Интернет через выделенную линию связи. Сумма — 25 000 руб.

Задание 2

Скачать из Интернета прайс-лист любой компьютерной фирмы (например, https://kiparis-crimea.ru) **и на его основе подобрать комплектующие для компьютера, предназначенного для решения определенного круга задач** (см. варианты ниже).

При выборе компонент компьютера необходимо уложиться в заданную сумму. Для подбора различных вариантов решения указанной задачи использовать табличный процессор (электронные таблицы). Все компоненты должны стыковаться с материнской платой по интерфейсу подключения и пропускной способности.

Вариант 1

Домашний компьютер. Компьютером будет пользоваться в основном ребенок 11 лет. Предполагается, что он будет использовать его для компьютерных игр и для учебы. Сумма, которой располагают родители, — 25 тыс. руб.

Вариант 2

Офисный компьютер. Компьютер будет использоваться в основном для подготовки и печати документов и выхода в Интернет. Он должен также входить в состав локальной сети фирмы. Сумма, которой располагает фирма, — 20 тыс. руб.

Вариант 3

Компьютер, предназначенный для рекламного агентства. Компьютер будет использоваться для работы с графическими приложениями и иногда для видеомонтажа небольших рекламных роликов. Сумма, которой располагает агентство, — 35 тыс. руб.

Примечание. Для видеомонтажа необходимо наличие на материнской плате интерфейса IEEE 1394 для подключения видеокамеры, а также два жестких диска.

Вариант 4

Учебный компьютер. Компьютер будет использоваться в учебном процессе и должен входить в локальную сеть школы. Сумма, которой располагает школа, — 20 тыс. руб.

Вариант 5

Домашний компьютер. Заказчик будет использовать компьютер для выхода в Интернет, просмотра видеофильмов, компьютерных игр, а также создания любительских фонограмм. Сумма, которой располагает заказчик, — 35 тыс. руб.

Вариант 6

Компьютер, предназначенный для работы Web- мастера. Заказчик будет использовать компьютер для выхода в Интернет и создания сайтов. При создании сайтов будет необходимо сканировать рисунки и фотографии. Сумма, которой располагает заказчик, — 25 тыс. руб.

Вариант 7

Учебный компьютер. Компьютер будет использоваться для обучения начальному пользовательскому курсу (Windows, Microsoft Office), включая печать документов, а также работе с пакетами CorelDraw, Photoshop и 3Dmax. Сумма, которой располагает учебный центр, — 25 тыс. руб.

Вариант 8

Компьютер, который будет использоваться профес-сиональным программистом (Delphi, базы данных и т. д.). Программист будет также использовать сетевой принтер (который уже есть в офисе, поэтому в комплект его включать не нужно). Сумма, которой располагает фирма, — 22 тыс. руб.

Вариант 9

Компьютер, который будет использоваться на телестудии для создания рекламных роликов. Сумма, которой располагает телестудия, — 35 тыс. руб.

Примечание. Для видеомонтажа необходимо наличие на материнской плате интерфейса IEEE 1394 для подключения видеокамеры, а также два жестких диска.

Вариант 10

Домашний компьютер. Компьютер должен быть предназначен в основном для просмотра видеофильмов с выводом на экран телевизора, компьютерных игр, прослушивания музыки. Сумма, которой располагает заказчик, — 27 тыс. руб.

4. Подведение итогов урока

Оценки за урок, следующие _____

5. Домашнее задание

Записываем домашнее задание: §§7-11
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 19
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Алгоритмы и величины. Структура алгоритмов. Структурное программирование.
Тип урока: Урок-лекция.
Цели урока:
Образовательная: формирование знаний учащихся об алгоритмах.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала
Показ видеоуроков с комментариями (15. Алгоритмы, величины, структура алгоритмов)

Этапы решения задачи на компьютере

Работа по решению любой задачи с использованием компьютера делится на следующие этапы:

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

Часто эту последовательность называют ***технологической цепочкой решения задачи на компьютере***. Непосредственно к программированию в этом списке относятся пункты **3, 4, 5**.

На **этапе постановки задачи** должно быть четко определено, ***что дано и что требуется найти***. Здесь очень важно определить полный набор исходных данных, необходимый для решения задачи.

Второй этап — формализация задачи. Здесь чаще всего задача переводится на язык математических формул, уравнений, отношений. Если решение задачи требует математического описания какого-то реального объекта, явления или процесса, то формализация равносильна получению соответствующей ***математической модели***.

Третий этап — построение алгоритма. Опытные программисты часто сразу пишут программы на языках программирования, не прибегая к каким-либо специальным способам описания алгоритмов (блок-схемам, псевдокодам). Однако в учебных целях полезно использовать эти средства, а затем переводить полученный алгоритм на язык программирования.

Первые три этапа — это работа без компьютера. Дальше следует собственно программирование на определенном языке в определенной системе программирования. Последний (шестой) этап — это уже использование разработанной программы в практических целях. Выполнение учебных заданий на программирование обычно заканчивается пятым этапом, т. е. доказательством правильности составленной программы.

Таким образом, программист должен обладать следующими знаниями и навыками:
• уметь строить алгоритмы;
• знать языки программирования;
• уметь работать в соответствующей системе программирования.

Основой программистской грамотности является развитое алгоритмическое мышление.

Понятие алгоритма

Одним из фундаментальных понятий в информатике является понятие алгоритма. Сам термин «алгоритм» пришел из математики. Это слово происходит от «Algorithmi» — латинского написания имени Мухамеда аль-Хорезми (787-850 гг.), выдающегося математика средневекового Востока. В XII веке был осуществлен латинский перевод его математического трактата, из которого европейцы узнали о десятичной позиционной системе счисления и правилах арифметики многозначных чисел. Именно эти правила в то время называли алгоритмами. Сложение, вычитание, умножение «столбиком», деление «уголком» многозначных чисел — вот первые алгоритмы в математике. Правила алгебраических преобразований, вычисление корней уравнений также можно отнести к математическим алгоритмам.

В наше время понятие алгоритма трактуется шире.

Алгоритм — это последовательность команд управления каким-либо исполнителем.

В школьном курсе информатики с понятием алгоритма, с методами построения алгоритмов ученики впервые знакомятся на примерах учебных исполнителей: Робота, Черепашки, Чертежника и др. В учебнике для 9 класса описан графический исполнитель — ГРИС. Эти исполнители ничего не вычисляют. Они создают рисунки на экране, перемещаются в лабиринтах, перетаскивают предметы с места на место. Таких исполнителей принято называть ***исполнителями, работающими в обстановке***.

В разделе информатики под названием «Программирование» изучаются методы программного управления работой компьютера. Следовательно, в качестве исполнителя выступает компьютер. Он работает с величинами — различными информационными объектами: числами, символами, кодами и пр. Поэтому алгоритмы, предназначенные для управления компьютером, принято называть ***алгоритмами работы с величинами***.

Данные и величины

Совокупность величин, с которыми работает компьютер, принято называть данными.

По отношению к программе данные делятся на исходные, результаты (окончательные данные) и промежуточные данные, которые получаются в процессе вычислений (рис. 3.1).

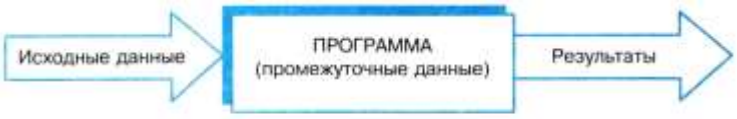


Рис. 3.1. Уровни данных относительно программы

Например, при решении квадратного уравнения: $ax^2 + bx + c = 0$ исходными данными являются коэффициенты a, b, c; результатами — корни уравнения x_1, x_2; промежуточными данными — дискриминант уравнения: $D = b^2 - 4ac$.

Для успешного освоения программирования необходимо усвоить следующее правило: ***всякая величина занимает свое определенное место в памяти компьютера***. Иногда говорят — ячейку памяти. Хотя термин «ячейка», с точки зрения архитектуры современных компьютеров, несколько устарел, однако в учебных целях его удобно использовать.

У всякой величины имеются три основных свойства: **имя, значение и тип**. На уровне команд процессора величина идентифицируется адресом ячейки памяти, в которой она хранится. В алгоритмах и языках программирования величины делятся на константы и переменные. Константа — неизменная величина, и в алгоритме она представляется собственным значением, например: 15, 34.7, 'k', true. Переменные величины могут изменять свои значения в ходе выполнения программы и представляются символическими именами — идентификаторами, например: X, S2, cod15. Любая константа или переменная занимают ячейку памяти, а значение этих величин определяется двоичным кодом в этой ячейке.

Теперь о типах величин — **типах данных**. С понятием типа данных вы уже встречались, изучая в курсе информатики основной школы электронные таблицы и базы данных. Это понятие является фундаментальным для программирования.

В каждом языке программирования существует своя концепция типов данных, своя система типов. Однако в любой язык входит минимально необходимый набор основных типов данных, к которому относятся целый, вещественный, логический и символьный типы. С типом величины связаны три ее свойства: множество допустимых значений, множество допустимых операций, форма внутреннего представления. В таблице 3.1 представлены эти свойства основных типов данных.

Типы констант определяются по контексту (т. е. по форме записи в тексте), а типы переменных устанавливаются в описаниях переменных.

Есть еще один вариант классификации данных: классификация по структуре. Данные делятся на **простые и структурированные**. Для простых величин (их еще называют скалярными) справедливо утверждение: одна величина — одно значение. Для структурированных: одна величина — множество значений.

К структурированным величинам относятся массивы, строки, множества и др.

Компьютер — исполнитель алгоритмов.

Как известно, всякий алгоритм (программа) составляется для конкретного исполнителя, в рамках его системы команд. О каком же исполнителе идет речь в теме «Программирование обработки информации»? Ответ очевиден: исполнителем является компьютер. Точнее говоря, исполнителем является комплекс: компьютер + система программирования (СП). Программист составляет программу на том языке, на который ориентирована СП. Схематически это изображено на рис. 3.2, где входным языком исполнителя является язык программирования Паскаль.

Таблица 3.1

Тип	Значения	Операции	Внутреннее представление
Целый	Целые положительные и отрицательные числа в некотором диапазоне. Примеры: 23, –12, 387	Арифметические операции с целыми числами: +, –, *, целочисленное деление и остаток от деления. Операции отношений (<, >, = и др.)	Формат с фиксированной запятой
Вещественный	Любые (целые и дробные) числа в некотором диапазоне. Примеры: 2.5, –0.01, 45.0, $3.6 \cdot 10^9$	Арифметические операции: +, –, *, /. Операции отношений	Формат с плавающей запятой
Логический	true (истина) false (ложь)	Логические операции: И (and), ИЛИ (or), НЕ (not). Операции отношений	1 бит: 1 — true; 0 — false
Символьный	Любые символы компьютерного алфавита. Примеры: 'a', '5', '+', 'S'	Операции отношений	Коды таблицы символьной кодировки. 1 символ — 1 байт

Рис. 3.2. Взаимодействие программиста с компьютером

Независимо от того, на каком языке программирования будет написана программа, **алгоритм решения любой задачи на компьютере может быть составлен из команд:**

- присваивания;
- ввода;
- вывода;
- обращения к вспомогательному алгоритму (подпрограмме);
- цикла;
- ветвления.

Эдсгер В. Дейкстра (1930–2002)

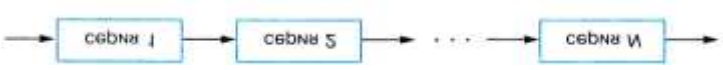
Для описания алгоритмов в дальнейшем мы будем использовать блок-схемы и учебный Алгоритмический язык, применяемый в школьном курсе информатики.

Базовые алгоритмические структуры

В 1969 году известным голландским ученым - программистом Э. В. Дейкстрой было доказано, что **алгоритм для решения любой логической задачи можно составить только из структур следование, ветвление, цикл**. Их называют **базовыми алгоритмическими структурами**. Методика программирования, основанная на этой теореме, называется ***структурным программированием***.

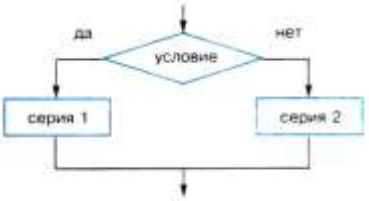
С базовыми алгоритмическими структурами вы познакомились, изучая информатику в 9 классе. Там же для описания структур алгоритмов были использованы два способа: блок-схемы и учебный Алгоритмический язык (АЯ). Еще раз покажем, как изображаются базовые структуры в схемах алгоритмов и как они описываются на АЯ.

Следование — это линейная последовательность действий (рис. 3.3).



В программе на Паскале серия — это либо один отдельный оператор, либо составной оператор: последовательность операторов, заключенная в операторные скобки. Например, в языке Паскаль операторными скобками являются служебные слова Begin и End.

Ветвление — алгоритмическая альтернатива. Управление передается одному из двух блоков в зависимости от истинности или ложности условия. Затем происходит выход на общее продолжение. Вот как изображается ветвление на блок-схеме и АЯ (рис. 3.4).

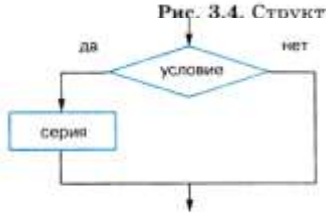


если <условие>
то <серия 1>
иначе <серия 2>
все

Рис. 3.4. Структура «ветвление»

Условие представляет собой утверждение, которое может быть либо истинным, либо ложным. Такое утверждение называется логическим выражением.

Неполная форма ветвления имеет место, когда на ветви «нет» пусто (рис. 3.5).



если <условие>
то <серия>
все

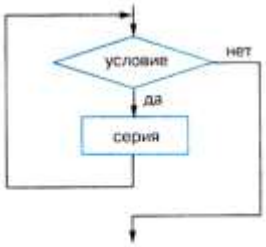
Рис. 3.5. Неполное ветвление

Цикл — повторение некоторой группы действий по условию. Различают два типа цикла. Первый — цикл с предусловием: **цикл-пока** (рис. 3.6).

Пока условие истинно, выполняется серия, образующая тело цикла.

Второй тип циклической структуры — цикл с постусловием: **цикл-до** (рис. 3.7).

Здесь тело цикла предшествует условию цикла. Тело цикла повторяет свое выполнение, если условие ложно. Повторение прекращается, когда условие становится истинным.



если <условие>
нц
<серия>
все

Рис. 3.6. Структура «цикл-пока»

повторить
<серия>
до <условие>

Рис. 3.7. Структура «цикл-до»

Теоретически необходимым и достаточным является лишь первый тип цикла — цикл с предусловием. Любой циклический алгоритм можно построить с его помощью. Это более общий вариант цикла, чем цикл-до. В самом деле, тело цикла-до хотя бы один раз обязательно выполнится, так как проверка условия происходит после завершения его выполнения. А для цикла-пока возможен такой вариант, когда тело цикла не выполнится ни разу. Поэтому в любом языке программирования можно было бы ограничиться только циклом-пока. Однако в ряде случаев применение цикла-до оказывается более удобным, и поэтому он используется.

Иногда в литературе структурное программирование называют *программированием без GOTO — **оператора безусловного перехода***. Действительно, при таком подходе нет места безусловному переходу. Неоправданное использование в программе оператора GOTO лишает ее структурности, а значит, всех связанных с этим положительных свойств: прозрачности и надежности алгоритма. Хотя во всех процедурных языках программирования этот оператор присутствует, однако, с точки зрения структурного подхода, его употребления следует избегать.

Комбинации базовых структур

Сложный алгоритм состоит из соединенных между собой базовых структур. Соединяться эти структуры могут двумя способами: ***последовательным и вложенным***.

Если блок, составляющий тело цикла, сам является циклической структурой, то имеют место вложенные циклы. В свою очередь, внутренний цикл может иметь внутри себя еще один цикл и т. д. В связи с этим ***вводится представление о глубине вложенности циклов***. Точно так же и ветвления могут быть вложенными друг в друга.

Структурный подход требует соблюдения стандарта в изображении блок-схем алгоритмов. Чертить их нужно так, как это делалось во всех приведенных примерах. Каждая базовая структура должна иметь один вход и один выход. Нестандартно изображенная блок-схема плохо читается, теряется наглядность алгоритма. Несколько примеров структурных блок-схем алгоритмов приведены на рис. 3.8 (вместо «да», «нет» здесь использованы знаки « + » и «-», У — <условие>, С — <серия>).

Такие блок-схемы легко читаются. Их структура хорошо воспринимается визуально. Структуре каждого алгоритма можно дать название. Приведенные блок-схемы можно охарактеризовать следующим образом (в порядке номеров).

1. Вложенные ветвления. Глубина вложенности равна единице.
2. Цикл с вложенным ветвлением.
3. Вложенные циклы-пока. Глубина вложенности — 1.
4. Ветвление с вложенной последовательностью ветвлений на положительной ветви и с вложенным циклом-пока на отрицательной ветви.
5. Следование ветвления и цикла-до.
6. Вложенные циклы. Внешний — цикл-пока, внутренний — цикл-до.

Наглядность структуре описания алгоритма на АЯ придает структуризация внешнего вида текста.

Основной используемый для этого прием — сдвиги строк, которые должны подчиняться следующим правилам:

• конструкции одного уровня вложенности записываются на одном вертикальном уровне (начинаются с одной позиции в строке);
• вложенная конструкция записывается смещенной по строке на несколько позиций вправо относительно внешней для нее конструкции.

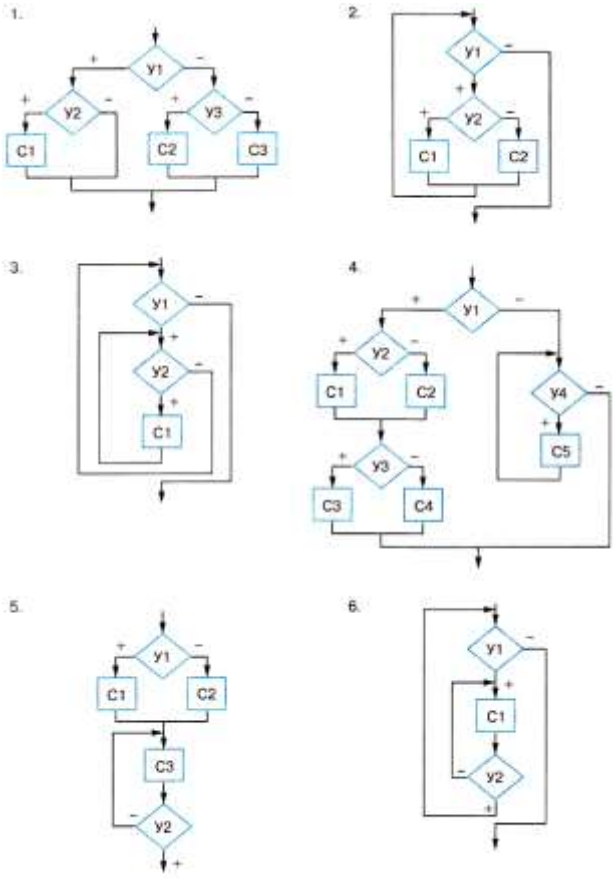


Рис. 3.8. Структурные схемы алгоритмов

Для приведенных на рис. 3.8 блок-схем структура текста на АЯ должна быть следующей:

```
1.
если <У1>
то если <У3>
   то  <С1>
   все
иначе если <У2>
        то <С2>
        иначе <С3>
        все
все
```

```
2.
пока <У1>
нц
   если <У2>
   то <С1>
   иначе <С2>
   все
кц
```

```
3.
пока <У1>
нц
   пока <У2>
   нц
      <С1>
   кц
кц
```

```
4.
если <У1>
то
   если <У2>
   то <С1>
   иначе <С2>
   все
   если <У3>
   то <С3>
   иначе <С4>
   все
иначе
   пока <У4>
   нц
      <С5>
   кц
все
```

```
5.
если <У1>
то <С1>
иначе <С2>
все
повторять
   <С3>
до <У2>
```

```
6.
пока <У1>
нц
   повторять
      <С1>
   до <У2>
кц
```

Такой же способ структуризации используется и в текстах программ (например, на Паскале).

Структурное программирование — это не только форма описания алгоритма и программы, но это еще и ***способ мышления программиста***. Размышляя над алгоритмом, нужно стремиться составлять его из стандартных структур. Если использовать строительную аналогию, то структурная методика построения алгоритма подобна сборке здания из стандартных секций, в отличие от складывания по кирпичику.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§12-14, задания №7 (§12) и №3,4 (§13) письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 17
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Элементы языка Паскаль, типы данных, операции, функции, выражения. Оператор присваивания, ввод и вывод данных.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала
Показ видеоуроков с комментариями (17. Язык структурного программирования Паскаль. Его элементы и типы данных, 18. Операции. Функции. Выражения)

Алфавит. Алфавит языка состоит из множества символов, включающих в себя буквы, цифры и специальные символы.

Латинские буквы: от ***A*** до ***Z*** (заглавные) и от ***a*** до ***z*** (строчные).

Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Специальные символы: + - * / = < > [ ] . , ( ) : ; { } ^ @ $ #.

Следующие комбинации специальных символов являются едиными символами (их нельзя разделять пробелами):

:= знак присваивания;
<= меньше или равно;
>= больше или равно;
(* *) ограничители комментариев (наряду с { });
<> не равно;
(. .) эквивалент [ ].

Пробелы — символ пробела (код ASCII 32) и все управляющие символы кода ASCII (от 0 до 31).

Служебные слова. К спецсимволам относятся и служебные слова, смысл которых определен однозначно. Служебные слова не могут быть использованы для других целей. С точки зрения языка, они являются едиными элементами алфавита. Вот некоторые служебные слова: **Program, Var, array, If, Do, While** и др.

Идентификаторы. Идентификатором называется символическое имя определенного программного объекта. Такими объектами являются: имена констант, переменных, типов данных, процедур и функций, программ. ***Идентификатор — это любая последовательность букв и цифр, начинающаяся с буквы***. К буквам приравнивается также знак подчеркивания. Длина идентификатора может быть произвольной, но значащими являются только первые 63 символа.

Комментарии. Следующие конструкции представляют собой комментарии и поэтому пропускаются компилятором:
{любой текст, не содержащий символ "фигурная скобка"}
(любой текст, не содержащий символы "звездочка, круглая скобка"*)*
//последующий текст до конца строки

Буквы русского алфавита употребляются только в комментариях, символьных и текстовых константах.

Концепция типов данных в Паскале

Концепция типов данных является одной из центральных в любом языке программирования. Как вы знаете, с **типом величины** *связаны три ее свойства: форма внутреннего представления, множество принимаемых значений и множество допустимых операций*.

Паскаль характеризуется большим разнообразием типов данных, отраженным на рис. 3.10.

Рис. 3.10. Система типов данных Паскаля

Тип данных называется **порядковым**, если он состоит из счетного количества значений, которые можно пронумеровать. Отсюда следует, что на этом множестве значений существуют понятия «следующий» и «предыдущий».

В стандартном Паскале Вирта отсутствовал строковый тип. Он был внесен в Турбо Паскаль. Кроме того, в Турбо Паскале целые и вещественные — это группы типов. В старших версиях Турбо Паскаля появился процедурный тип и тип «объект».

Каждый тип имеет свой идентификатор. В таблице 3.2 представлена информация о простых типах данных, определенных в Турбо Паскале и последующих диалектах языка. Для вещественных типов в скобках указано количество сохраняемых значащих цифр мантиссы в десятичном представлении числа.

Типы пользователя. Одна из принципиальных возможностей Паскаля состоит в том, что пользователю разрешается определять свои типы данных. Типы пользователя всегда базируются на стандартных типах данных Паскаля. Для описания типов пользователя в Паскале существует раздел типов, начинающийся со служебного слова ***Type***. К простым типам пользователя относятся перечислимый и интервальный типы данных.

Таблица 3.2. **Типы данных**

Имя типа	Длина в байтах	Диапазон (множество) значений	Десятичных цифр в мантиссе
Целочисленные типы			
Integer	2	–32768..32767	
Byte	1	0..255	
Word	2	0..65535	
Shortint	1	–128..127	
Longint	4	–2147483648..2147483647	
Вещественные типы			
Real	6	$2{,}9\cdot10^{-39} - 1{,}7\cdot10^{38}$	11–12
Single	4	$1{,}5\cdot10^{-45} - 3{,}4\cdot10^{38}$	7–8
Double	8	$5\cdot10^{-324} - 1{,}7\cdot10^{308}$	15–16
Extended	10	$3{,}4\cdot10^{-4932} - 1{,}1\cdot10^{4932}$	19–20
Логический тип			
Boolean	1	true, false	
Символьный тип			
Char	1	Все символы 8-разрядной кодировки	

Перечислимый тип задается непосредственно перечислением (списком) всех значений, которые может принимать переменная данного типа:

Type *<имя типа> = (<список значений>)*

Определенное таким образом имя типа затем используется для описания переменных. Например:

```
Type Gaz = (C, O, N, F);
     Metal = (Fe, Co, Na, Cu, Zn);
Var  G1, G2, G3: Gaz;
     Met1, Met2: Metall;
     Day: (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
```

Здесь ***Gaz*** и ***Metal*** — имена перечислимых типов, которые ставятся в соответствие переменным ***Gl, G2, G3*** и ***Met 1, Met2***. Переменной ***Day*** назначается перечислимый тип, которому не присвоено имени.

Значения, входящие в перечислимый тип, являются константами. Действия над ними подчиняются правилам, применимым к константам. Каждое значение в перечислимом типе занимает в памяти 2 байта, поэтому число значений этого типа не должно превышать 65 535.

Перечислимый тип — упорядоченное множество. Его элементы пронумерованы, начиная от 0 в порядке следования в описании.

Структурные типы. Особенностью Паскаля является то, что в нем структуры данных рассматриваются как типы — структур-

В программе, в которой присутствует данное выше описание переменной ***Day***, возможен такой фрагмент:

```
If Day=Sun Then Write('Ура! Сегодня выходной!');
```

Ограниченный тип задается как упорядоченное ограниченное подмножество некоторого порядкового типа:

<константа 1>..<константа 2>

Порядковый номер первой константы не должен превышать номера второй константы в соответствующем базовом типе.

При исполнении программы автоматически контролируется принадлежность значений переменной ограниченного типа установленному диапазону. При выходе из диапазона исполнение программы прерывается.

Пример

```
Type  Numbers = 1..31;
      Alf = 'A'..'Z';
Var   Data: Numbers;
      Bukva: Alf;
```

Структурные типы. Особенностью Паскаля является то, что в нем структуры данных рассматриваются как типы — структурные типы данных. Одна величина простого типа представляет собой одно значение: целое число, вещественное число, символ и пр. Одна величина структурного типа представляет собой совокупность множества значений; примеры — числовой массив, символьная строка и пр.

Автор Паскаля Вирт придавал большое значение разнообразию типов данных в языке программирования. В своей книге «Алгоритмы и структуры данных» он подчеркивает зависимость алгоритма решения задачи от способа организации данных в программе. Удачно выбранный способ организации данных упрощает алгоритм решения задачи.

Арифметические операции

К числовым типам данных относятся группы вещественных и целочисленных типов. К ним применимы арифметические операции и операции отношений. Операции над данными бывают унарными (применимые к одному операнду) и бинарными (применимые к двум операндам).

Унарная арифметическая операция в Паскале одна. Это операция изменения знака. ***Ее формат:***

-<величина>

Бинарные арифметические операции стандартного Паскаля описаны в табл. 3.3. В ней символ «**I**» обозначает целые типы, символ «**R**» — вещественные типы.

Таблица 3.3. **Бинарные операции Паскаля**

Знак	Выражение	Типы операндов	Тип результата	Операция
+	A+B	R, R I, I I, R; R, I	R I R	Сложение
-	A-B	R, R I, I I, R; R, I	R I R	Вычитание
*	A*B	R, R I, I I, R; R, I	R I R	Умножение
/	A/B	R, R I, I I, R; R, I	R R R	Вещественное деление
div	A div B	I, I	I	Целочисленное деление
mod	A mod B	I, I	I	Остаток от целочисленного деления

Стандартные функции и процедуры

В Паскале существует большое количество стандартных функций и процедур, к которым программист может обращаться в своих программах. Наиболее часто используются математические функции, например: *sqrt(x)* — квадратный корень, *abs (x)* — абсолютная величина, *sin (x)* и др. Часто используемые стандартные процедуры: *Read(...)* — процедура ввода, *Write (...)* — процедура вывода данных.

Стандартные функции и процедуры являются внешними подпрограммами по отношению к вызывающей их программе. Они объединены в модули, которые подключаются к основной программе и становятся доступными для использования. Наиболее часто используемые подпрограммы объединены в модуль под названием SYSTEM. Этот модуль подключается к программе автоматически.

Таблица 3.4. **Стандартные математические функции Паскаля**

Обращение	Тип аргумента	Тип результата	Функция
Pi	—	R	Число π = 3,14159265358E+00
abs(x)	I, R	I, R	Модуль аргумента
arctan(x)	I, R	R	Арктангенс (в радианах)
cos(x)	I, R	R	Косинус (в радианах)
exp(x)	I, R	R	e^x — экспонента
frac(x)	I, R	R	Дробная часть *x*
int(x)	I, R	R	Целая часть *x*
ln(x)	I, R	R	Натуральный логарифм
random	—	R	Псевдослучайное число в интервале [0, 1)
random(x)	I	I	Псевдослучайное число в интервале [0, *x*)
round(x)	R	I	Округление до ближайшего целого
sin(x)	I, R	R	Синус (в радианах)
sqr(x)	I, R	I, R	Квадрат *x*
sqrt(x)	I, R	R	Квадратный корень
trunc(x)	R	I	Ближайшее целое, не превышающее *x* по модулю

Таблица 3.4 содержит описания стандартных математических функций Паскаля.

Для подключения других модулей необходимо в начале программы (после заголовка) записать строку:

Uses *<имя модуля>*

Для управления символьным выводом на экран используется стандартный модуль CRT. К программе он подключается командой:

Uses *CRT*

В дальнейшем из этого модуля мы будем использовать *процедуру очистки экрана* для символьного вывода, обращение к которой производится оператором *ClrScr*.

Арифметические выражения

Арифметическое выражение задает порядок выполнения действий над числовыми величинами. Арифметические выражения содержат числовые константы и переменные, арифметические операции, функции, круглые скобки. Одна константа или одна переменная — простейшая форма арифметического выражения.

Например, рассмотрим математическое выражение:

$$\frac{2a+\sqrt{0{,}5\sin(x+y)}}{0{,}2c-\ln(x-y)}.$$

На Паскале оно выглядит так:

*(2*A + Sqrt(0.5*sin(X + Y))) / (0.2*C - ln(X - Y))*

Для того чтобы правильно записывать арифметические выражения, нужно соблюдать следующие правила.

1. Все символы пишутся в строчку на одном уровне. Проставляются все знаки операций (нельзя пропускать знак *).

2. Не допускаются два следующих подряд знака операций. (Нельзя: *A+-B; можно: A+ (-B)*.)

3. Операции с более высоким приоритетом выполняются раньше операций с меньшим приоритетом. Порядок убывания приоритетов:

вычисление функции;

унарная операция смены знака (-);

*, /, div, mod;

+, -.

4. Несколько записанных подряд операций одинакового приоритета выполняются последовательно слева направо.

5. Часть выражения, заключенная в скобки, вычисляется в первую очередь. (Например, в выражении *(A+B) * (C-D)* умножение производится после сложения и вычитания.)

Не следует записывать выражения, не имеющие математического смысла, например: деление на нуль, логарифм отрицательного числа и т. п.

Пример. Цифрами сверху указан порядок выполнения операций:

```
     1   7   4 5  3       6   2     12  11 10   8      9
(1+y)*(2*x+sqrt(y)-(x+y))/(y+1/(sqr(x)-4))
```

Данное арифметическое выражение (на Паскале) соответствует следующему математическому выражению:

$$(1+y)\frac{2x+\sqrt{y}-(x+y)}{y+\dfrac{1}{x^2-4}}.$$

В Паскале нет операции или стандартной функции возведения числа в произвольную степень. Для вычисления xy рекомендуется поступать следующим образом:

а) если *y* — целое положительное значение, то его степень вычисляется через умножение; например x^3 —> x*x*x; большие степени следует вычислять умножением в цикле;

б) если *y* — целое отрицательное число, то степень вычисляется так: $x^y = (1/x)^{|y|}$; а при *y* = 0: $x^0 = 1$.

в) если *y* — вещественное значение, не равное нулю, то используется следующая математическая формула: $x^y = e^{y1n(-x)}$ На Паскале получим арифметическое выражение:

*exp(Y*ln(x))*

Очевидно, что в этом случае не допускается нулевое или отрицательное значение x. Для целого y такого ограничения нет.

Пример

$$\sqrt[3]{a+1}=(a+1)^{\frac{1}{3}}.$$

На Паскале это выражение выглядит так:

*exp(1/3*ln(A+1))*

Выражение имеет целочисленный тип, если в результате его вычисления получается величина целочисленного типа. Выражение имеет вещественный тип, если результатом его вычисления является вещественная величина.

Присваивание — это действие, в результате которого переменная величина получает определенное значение. В программе на Паскале существуют три способа присваивания значения переменной:

1) оператор присваивания;
2) оператор ввода;
3) передача значения через параметры подпрограммы.

Оператор присваивания имеет следующий формат:

<переменная>:=<выражение>

Например:

```
1) x:=2*a+sqrt(b);
2) b:=(x>y)  and  (k<>0);
```

Сначала вычисляется выражение, затем полученное значение присваивается переменной. В первом примере приведен ***арифметический оператор присваивания***. Здесь ***x*** — переменная вещественного типа. Во втором примере — ***логический оператор присваивания***. Здесь ***b*** — переменная типа *Boolean*.

Типы переменной и выражения должны совпадать. Из этого правила есть одно исключение: переменной вещественного типа можно присваивать значение целочисленного выражения. В таком случае значение целого числа преобразуется к формату с плавающей запятой и присвоится вещественной переменной.

Ввод и вывод данных

Под вводом понимается передача данных с внешнего устройства компьютера в оперативную память. При выводе данные передаются из оперативной памяти на внешнее устройство (рис. 3.11).

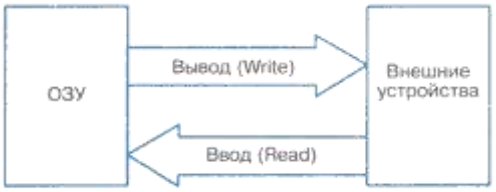


Рис. 3.11. Ввод и вывод

Операция ввода называется чтением и выполняется с помощью оператора ***Read***. Вывод называется записью, и для его выполнения используется оператор ***Write***.

К внешним устройствам относятся устройства ввода и вывода (клавиатура, монитор, принтер и др.) и устройства внешней памяти (магнитные и оптические диски, флеш-память и др.). Данные на внешних устройствах организованы в файлы.

Для внешних запоминающих устройств (ВЗУ) файл — это поименованная область памяти этого устройства. В файлы на ВЗУ можно записывать данные по команде Write и можно читать данные из файлов по команде ***Read***. На одном устройстве ВЗУ может храниться множество файлов одновременно. Правила именования файлов на ВЗУ определяются операционной системой. Имена для файлов, создаваемых пользователем, задает сам пользователь.

Устройства ввода с клавиатуры и вывода на экран монитора являются однофайловыми устройствами. Считается, что с клавиатурой связан один системный файл с именем ***INPUT***. Поэтому ввод с клавиатуры равнозначен чтению из файла ***INPUT***. С монитором связан системный файл, который называется ***OUTPUT***. Вывод на экран — это запись данных в файл ***OUTPUT*** .

Ввод с клавиатуры производится путем обращения к стандартной процедуре Read в следующем формате:

Read(<список ввода>)

Чтение происходит из системного файла ***INPUT***, всегда доступного для любой программы. Элементами списка ввода могут быть переменные символьного типа, числовых типов и строковые переменные.

Например:

Read(a, b, c, d)

При выполнении этого оператора происходит прерывание исполнения программы, после чего пользователь должен набрать на клавиатуре значения переменных a, Ъ, c, d, отделяя их друг от друга пробелами. При этом вводимые значения высвечиваются на экране. В конце нажимается клавиша Enter. Значения следует вводить в строгом соответствии с синтаксисом Паскаля.

Пример:

```
Var T: Real; J: Integer;   K: Char;
Begin
  Read(T, J, K);
```

Набираем на клавиатуре:

```
253.98 100 G [Enter]
```

Если в программе имеется несколько подряд идущих операторов Read, то данные для них можно вводить последовательно (на экране отражаются в одной строке) и лишь в конце ввода нужно нажать клавишу Enter.

Пример:

```
Var A, B: Integer;
    C, D: Real;
Begin
    Read(A, B);   Read(C, D);
```

Набираем на клавиатуре и видим на экране:

```
18758 34 2.62E-02 1.54E+01 [Enter]
```

Другой вариант оператора ввода с клавиатуры имеет вид:

```
ReadLn(<список ввода>)
```

Здесь слово ***«ReadLn»*** означает ***read line*** — «читать строку». Нажатие клавиши **Enter** в процессе ввода вырабатывает признак **«конец строки»**, и данные при выполнении следующего оператора ввода будут отражаться на экране с начала новой строки. Если в предыдущем примере заменить операторы ***Read*** на ***ReadLn***:

ReadLn(A, B); ReadLn(C, D) ;

то ввод значений будет происходить из двух строк, отраженных на экране:

18758 34 **[Enter]**

2.62E-02 1.54E+01 **[Enter]**

Вывод на экран производится по оператору обращения к стандартной процедуре:

Write(<*список вывода*>)

Здесь элементами списка вывода могут быть выражения различных типов (в частности, константы и переменные).

Например: Write ('Сумма A, ' + ', B, ' = ', A+B)

Если, например, A = 5 и B = 7, то на экране получим:

Сумма 5+7=12

При выводе на экран нескольких значений в строку они не отделяются друг от друга пробелами. Программист сам должен позаботиться о таком разделении. В приведенном примере предусмотрен пробел после слова «Сумма».

Второй вариант процедуры вывода на экран:

WriteLn(<*список вывода*>)

Write line — «писать строку». Его действие отличается от оператора ***Write*** тем, что после вывода последнего в списке значения происходит перевод курсора к началу следующей строки. Оператор ***WriteLn***, записанный без параметров, вызывает перевод строки.

В списке вывода могут присутствовать указатели форматов вывода (форматы). Формат определяет представление выводимого значения на экране. Формат отделяется от соответствующего ему элемента двоеточием. Если указатель формата отсутствует, то машина выводит значение по определенному правилу, предусмотренному по умолчанию.

Линейная программа. Следование — простейшая алгоритмическая структура. Программа, реализующая следование, называется линейной программой. В линейной программе могут присутствовать только операторы присваивания, ввода, вывода и обращения к процедурам. Заметим, что операторы Read и Write являются обращениями к стандартным процедурам Паскаля.

Одним из обязательных условий хорошего стиля программирования является организация диалога между компьютером и пользователем. Такое диалоговое взаимодействие называется ***интерактивным интерфейсом***.

Пример 1. Составим линейную программу, по которой в диалоге будут вводиться два целых числа и вычисляться их произведение.

```
Program Multiply;
Var A, B, AB: integer;
Begin
   Write('A= '); ReadLn(A);
   Write('B= '); ReadLn(B);
   AB:=A*B;
   Write(A, '*', B, '=', AB)
End.
```

Тестирование этой программы отразится на экране следующим образом.

A= 13

B= 28

13*28=364

Числа 13 и 28 вводятся пользователем с клавиатуры, всё остальное автоматически выводится по программе.

Пример 2. Дано натуральное трехзначное число. Требуется вычислить сумму его цифр. Например, если дано число 325, то в результате должно получиться: 3 + 2 + 5 = 10.

Сначала составим программу, а потом ее прокомментируем.

```
Program SumCifr;
Var X, Sum: integer;
Begin
  Write('Введите трехзначное число: '); ReadLn(X);
  Sum:=0;
  Sum:=Sum + X mod 10;
  X:=X div 10;
  Sum:=Sum + X mod 10;
  X:=X div 10;
  Sum:=Sum + X;
  Write('Сумма цифр = ', Sum)
End.
```

В этой программе использованы две операции целочисленной арифметики: ***div*** — целочисленное деление и ***mod*** — остаток от целочисленного деления (см. табл. 3.3). Остаток от деления на 10 (mod) выделяет младшую цифру числа, а целочисленное деление на 10 (div) отбрасывает младшую цифру.

№	Команда	X	Sum
1	readln(X)	325	–
2	Sum:=0		0
3	Sum:=Sum + X mod 10		5
4	X:=X div 10	32	
5	Sum:=Sum + X mod 10		7
6	X:=X div 10	3	
7	Sum:=Sum + X		10
8	Write('Сумма цифр = ', Sum)		10

Чтобы лучше понять работу программы, выполним ее трассировку. В курсе 9 класса вам уже приходилось строить трассировочные таблицы. Для программы ***SumCifr*** таблица будет выглядеть следующим образом:

Выполнение программы на компьютере приводит к такому же результату.

Заметим, что эту задачу можно решить с помощью всего одного оператора присваивания:

Sum: =X mod 10 + X div 10 mod 10 + X div 100

Проверьте самостоятельно.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§15-17, задания №1,2 (§16) и №4,6 (§17) письменно в тетради. Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 21
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №9 «Программирование линейных алгоритмов».
Тип урока: Урок-практикум.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Практическая работа

Ход работы

Задание

Для каждой вычислительной задачи составить программу, содержащую операторы ввода, вывода, присваивания.

1. Вычислить длину окружности и площадь круга одного и того же заданного радиуса ***R***.

2. Вычислить расстояние между двумя точками с данными координатами на плоскости (x_1, y_1) и (x_2, y_2).

3. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

4. Три сопротивления ***R1, R2, R3*** соединены параллельно. Найти сопротивление всей цепи.

5. Найти сумму членов арифметической прогрессии, если известны ее первый член, разность и число членов прогрессии.

6. Вычислить корни квадратного уравнения $ax^2 + bx + c = 0$ с заданными коэффициентами ***a, b и c*** (предполагается, что **a** ≠ 0 и что дискриминант уравнения неотрицателен).

7. Найти площадь равнобедренной трапеции с основаниями **a** и **b** и углом ***α*** при большем основании **a**.

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§15-17.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 22
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Логические величины, операции, выражения. Программирование ветвлений.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

С элементами математической логики вы уже встречались в курсе информатики основной школы, изучая способы записи запросов к базе данных и условной функции *ЕСЛИ* в электронных таблицах, основы алгоритмизации и программирования. Повторим основные понятия логики с целью дальнейшего углубления ваших знаний в использовании ее для программирования.

К числу основных понятий логики относятся: ***высказывание, логическая величина, логические операции, логические выражения и формулы.***

Высказывание (суждение) — это повествовательное предложение, в котором что-либо утверждается или отрицается. По поводу любого высказывании можно сказать, истинно оно или ложно.

Например, высказывание «На улице идет дождь» будет истинным или ложным в зависимости от состояния погоды в данный момент. Истинность высказывания «Значение А больше, чем В», записанного в форме неравенства: A > B, будет зависеть от значений переменных А и В.

Логические величины — понятия, выражаемые словами: ИСТИНА, ЛОЖЬ (true, false). Следовательно, *истинность высказываний выражается через логические величины*.

Логическая константа: ИСТИНА или ЛОЖЬ.

Логическая переменная: символически обозначенная логическая величина. Следовательно, если известно, что A, B, X, Y и др. — переменные логические величины, то, значит, они могут принимать значения только ИСТИНА или ЛОЖЬ.

Логическое выражение — простое или сложное высказывание. Сложное высказывание строится из простых с помощью логических операций (связок).

Логические операции

Конъюнкция (логическое умножение). В русском языке она выражается союзом И. В математической логике используются знаки & или ∧. **Конъюнкция — двухместная операция;** записывается в виде: A & B. Значением такого выражения будет ЛОЖЬ, если значение хотя бы одного из операндов ложно.

Дизъюнкция (логическое сложение). В русском языке этой связке соответствует союз ИЛИ. В математической логике она обозначается знаком **v. Дизъюнкция — двухместная операция;** записывается в виде: A v B. Значением такого выражения будет ИСТИНА, если значение хотя бы одного из операндов истинно.

Отрицание. В русском языке этой связке соответствует частица НЕ (в некоторых высказываниях применяется оборот «неверно, что ...»). **Отрицание — унарная (одноместная) операция;** записывается в виде: ¬ A или $\bar{A}$.

Правила выполнения рассмотренных логических операций отражены в следующей таблице, которая называется таблицей истинности логических операций (здесь И означает «истина», Л — «ложь»):

A	*B*	¬*A*	*A* & *B*	*A* ∨ *B*
И	И	Л	И	И
И	Л	Л	Л	И
Л	И	И	Л	И
Л	Л	И	Л	Л

Логическая формула — формула, содержащая лишь логические величины и знаки логических операций. Результатом вычисления логической формулы является ИСТИНА или ЛОЖЬ.

Последовательность выполнения операций в логических формулах определяется старшинством операций. В порядке убывания старшинства логические операции расположены так: ***отрицание, конъюнкция, дизъюнкция***. Кроме того, на порядок выполнения операций влияют скобки, которые можно использовать в логических формулах.

Например: (A & B) v (¬ A & B) v (¬ A & ¬ B).

Пример. Вычислить значение логической формулы:

¬ X & Y v X & Z,

если логические переменные имеют следующие значения: *X* = ЛОЖЬ, *Y* = ИСТИНА, *Z* = ИСТИНА.

Решение. Отметим цифрами сверху порядок выполнения операций в формуле:

$$\overset{1}{\neg} X \overset{2}{\&} Y \overset{4}{\vee} X \overset{3}{\&} Z.$$

Используя таблицу истинности, вычислим формулу по шагам:

1) ЛОЖЬ = ИСТИНА;
2) ИСТИНА & ИСТИНА = ИСТИНА;
3) ЛОЖЬ & ИСТИНА = ЛОЖЬ;
4) ИСТИНА v ЛОЖЬ = ИСТИНА.

Ответ: ИСТИНА.

Логические функции на области числовых значений

Алгебра чисел пересекается с алгеброй логики в тех случаях, когда приходится проверять принадлежность значений алгебраических выражений некоторому множеству. Например, принадлежность значения числовой переменной X множеству положительных чисел выражается через *высказывание*: «X больше нуля». Символически это записывается так: X > 0. В алгебре такое выражение называют неравенством. В логике — отношением.

Отношение X > 0 может быть истинным или ложным. Если X — положительная величина, то оно истинно, если отрицательная, то ложно. В общем виде отношение имеет следующую структуру:

< выражение 1 > < знак отношения > < выражение 2 >

Здесь выражения 1 и 2 — некоторые математические выражения, принимающие числовые значения. В частном случае выражение может представлять собой одну константу или одну переменную величину. Знаки отношений могут быть следующими:

= — равно;
≠ — не равно;
≥ — больше или равно;
≤ — меньше или равно;
> — больше;
< — меньше.

Например:

$$x = 5; \quad a + b \neq x - 1; \quad b^2 - 4ac \geq 0; \quad \sin(x) < x/2.$$

Итак, отношение — это простое высказывание, а значит, логическая величина. Оно может быть как постоянной: 5 > 0 — всегда ИСТИНА, 3 * 6 : 2 — всегда ЛОЖЬ; так и переменной: a < b, x + 1 = c - d. Если в отношение входят переменные числовые величины, то и значение отношения будет логической переменной.

Отношение можно рассматривать как логическую функцию от числовых аргументов. Например: F(x) = (x > 0) или P(x, y) = = (x < y). Аргументы определены на бесконечном множестве действительных чисел, а значения функции — на множестве, состоящем из двух логических величин: ИСТИНА, ЛОЖЬ.

Логические функции от числовых аргументов еще называют термином **предикат**. В алгоритмах предикаты играют роль условий, по которым строятся ветвления и циклы. Предикаты могут быть как

простыми логическими функциями, не содержащими логических операций, так и сложными, содержащими логические операции.

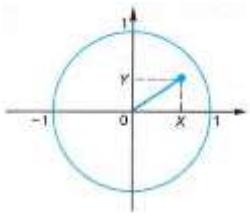


Рис. 3.12

Пример 1. Записать предикат (логическую функцию) от двух вещественных аргументов X и Y, который будет принимать значение ИСТИНА, если точка на координатной плоскости с координатами X и Y лежит внутри единичной окружности с центром в начале координат (рис. 3.12).

Из геометрических соображений понятно, что для всех точек, лежащих внутри единичной окружности, будет истинным значение следующей логической функции:

$F(X, У) = (X^2 + У^2 < 1)$.

Для значений координат точек, лежащих на окружности и вне ее, значение функции F будет ложным.

Пример 2. Записать предикат, который будет принимать значение ИСТИНА, если точка на координатной плоскости с координатами X и У лежит внутри кольца с центром в начале координат, и радиусами R1 и R2.

Поскольку значения R1 и R2 — переменные величины, искомая логическая функция будет иметь четыре аргумента: X, У, R1, R2. Возможны две ситуации:

1) $R1^2 < X^2 + У^2 < R2^2$ и R1 < R2: R1 — внутренний радиус, R2 — внешний радиус;

2) $R2^2 < X^2 + У^2 < R1^2$ и R2 < R1: R2 — внутренний радиус, R1 — внешний радиус.

Объединив дизъюнкцией оба этих утверждения и записав их по правилам алгебры логики, получим следующую логическую функцию:

$F(X, У, R1, R2) = (((X^2 + У^2) > R1^2) \& ((X^2 + У^2) < R2^2) \& R1 < R2) \vee (((X^2 + У^2) > R2^2) \& ((X^2 + У^2) < R1^2) \& R2 < R1)$.

Пример 3. Записать предикат, который будет принимать значение ИСТИНА, если точка на координатной плоскости с координатами X и У лежит внутри фигуры, ограниченной жирными линиями на рис. 3.13.

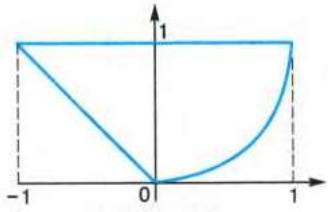


Рис. 3.13

Фигура ограничена тремя границами, описываемыми уравнениями:

У = -X — левая граница, линейная функция;

У = 1 — верхняя граница, константа;

$У = X^2$ — правая граница, парабола.

Рассматриваемая область есть пересечение трех полуплоскостей, описываемых неравенствами:

$$\begin{cases} Y > -X; \\ Y < 1; \\ Y > X^2. \end{cases}$$

Во внутренних точках все эти три отношения являются одно-временно истинными. Поэтому искомый предикат имеет вид:

$F(X, У) = (У > -X) \& (Y < 1) \& (У > X^2)$.

Логические выражения на Паскале

Уже говорилось о том, что в Паскале имеется логический тип данных.

Логические константы: *true* (истина), *false* (ложь).

Логические переменные: описываются с типом *Boolean*.

Операции отношения: осуществляют сравнение двух операндов и определяют, истинно или ложно соответствующее отношение между ними. Знаки операций отношения: = (равно), <> (не равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно).

Логические операции: *not* — отрицание, *and* — логическое умножение (конъюнкция), *or* — логическое сложение (дизъюнкция), *xor* — исключающее ИЛИ. Таблица истинности для этих операций (T — *true*; F — *false*):

A	B	not A	A and B	A or B	A xor B
T	T	F	T	T	F
T	F	F	F	T	T
F	T	T	F	T	T
F	F	T	F	F	F

Логическое выражение может состоять из логических констант и переменных, отношений, логических операций. Логическое выражение принимает значение true или false.

Например, логическая формула ¬ X & У v X & Z на Паскале запишется в виде следующего логического выражения:

not X **and** Y **or** X **and** Z,

где *X, Y, Z* — переменные типа *Boolean*.

Логические операции располагаются в следующем порядке по убыванию старшинства (приоритета): 1) **not**, 2) **and**, 3) **or, xor**. Операции отношения имеют самый низкий приоритет. Поэтому если операндами логической операции являются отношения, то их следует заключать в круглые скобки. Например, математическому неравенству $1 \le X \le 50$ соответствует следующее логическое выражение:

(1 <= X) **and** (X <= 50)

Логическая функция **odd(x)** принимает значение *true*, если значение целочисленного аргумента ***x*** является нечетным, иначе — *false*.

Для правильной записи сложного логического выражения (предиката) нужно учитывать относительные приоритеты арифмети-ческих, логических операций и операций отношений, поскольку все они могут присутствовать в логическом выражении. По убыванию приоритета операции располагаются в следующем порядке.

1. Арифметические операции:

- (минус унарный)
*, /
+, -

2. Логические операции:

not
and
or, xor

3. Операции отношения:

=, <>, >, <, >=, <=

Еще раз обратите внимание, что в логическом выражении, соответствующем предикату из примера 3:

(Y > -X) **and** (Y < 1) **and** (Y > X * X),

операции отношения заключены в скобки, поскольку они младше логических операций, а выполняться должны раньше.

Программирование ветвлений

На уроках "Программирование линейных алгоритмов" был показан способ отображения ветвления (полного и неполного) на блок-схеме и учебном Алгоритмическом языке. Алгоритмическая структура ветвления программируется в Паскале с помощью условного оператора ***If***. В 9 классе вы познакомились с этим оператором. Вспомним его формат.

Полное ветвление:

If < логическое выражение >
Then < оператор 1 >
Else < оператор 2 >

Неполное ветвление:

If < логическое выражение >
Then < оператор >

То, что в алгоритмах называется условием, в Паскале является *логическим выражением*, которое вычисляется в первую очередь. Если его значение равно true, то будет выполняться < оператор 1 > (после Then), если — false, то < оператор 2 > (после Else) для полной формы или оператор, сразу следующий после условного, для неполной формы (без Else). На ветвях может быть как простой оператор, так и составной — серия операторов в операторных скобках Begin, End.

Пример 1. По длинам трех сторон треугольника a, b, c требуется вычислить его площадь.

Для решения задачи используется формула Герона

$$\sqrt{p(p-a)(p-b)(p-c)},$$

где p = (a + b + c)/2 — полупериметр треугольника. Исходные данные должны удовлетворять основному соотношению для сторон треугольника — длина каждой стороны должна быть меньше суммы длин двух других сторон, и длины сторон не могут быть отрицательными величинами.

Имея возможность в одном условном операторе записывать достаточно сложные логические выражения, используя логические операции, мы можем сразу «отфильтровать» все варианты неверных исходных данных.

Пример 2. Требуется перевести пятибалльную оценку в ее на-именование: 5 — «отлично», 4 — «хорошо», 3 — «удовлетворительно», 2 — «неудовлетворительно».

Блок-схема алгоритма приведена на рис. 3.14.

Этот алгоритм имеет структуру вложенных ветвлений и может быть запрограммирован с использованием условного оператора **If** следующим образом:

```
Program Geron;
Var A, B, C, P, S : Real;
Begin
  WriteLn('Введите длины сторон треугольника: ');
  Write('a = '); ReadLn(A);
  Write('b= '); ReadLn(B);
  Write('c='); ReadLn(C);
  If (A>0) and (B>0) and (C>0) and (A+B>C)
     and (B+C>A) and (A+C>B)
  Then Begin
         P := (A+B+C)/2;
         S := Sqrt(P*(P-A)*(P-B)*(P-C));
         WriteLn('Площадь=', S)
       End
  Else WriteLn('Неверные исходные данные')
```

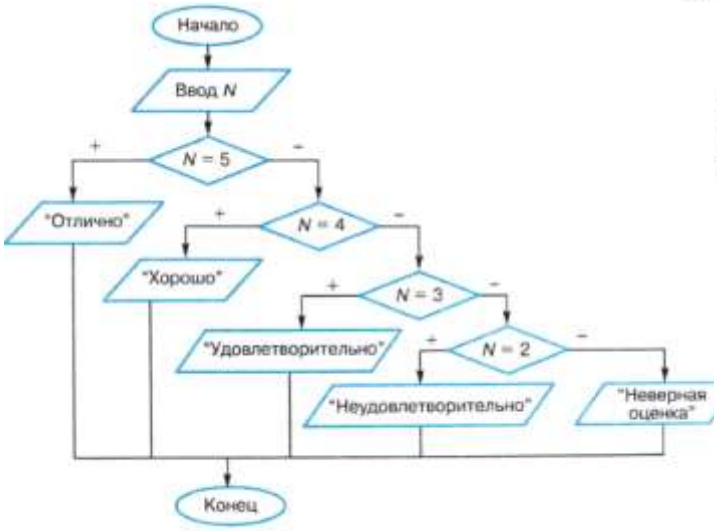


Рис. 3.14. Алгоритм перевода числовой оценки в словесную

```
Program Marks_1;
Var N: Integer;
Begin
  WriteLn('Введите оценку:');
  ReadLn(N);
  If N=5
  Then WriteLn('Отлично')
  Else If N=4
    Then WriteLn('Хорошо')
    Else If N=3
      Then WriteLn('Удовлетворительно')
      Else If N=2
        Then WriteLn('Неудовлетворительно')
        Else WriteLn('Неверная оценка')
End.
```

Пример 3. Решение рассмотренной в предыдущем примере задачи можно запрограммировать с помощью одного оператора выбора, имеющегося в языке Паскаль. Вот как будет выглядеть такая программа:

```
Program Marks_2;
Var N: Integer;
Begin
  WriteLn('Введите оценку:');
  ReadLn(N);
  Case N of
    5: WriteLn('Отлично');
    4: WriteLn('Хорошо');
    3: WriteLn('Удовлетворительно');
    2: WriteLn('Неудовлетворительно');
    Else WriteLn('Неверная оценка')
  End;
```

Оператор выбора имеет следующий формат:

Case < селектор > **of**
< список констант 1 > : < оператор 1 > ;
…
< список констант N > : < оператор N > ;
Else <оператор>
End

Здесь < селектор > — это выражение любого порядкового типа; < константа > — постоянная величина того же типа, что и селектор; < оператор > — любой простой или составной оператор.

Выполнение оператора выбора происходит так: вычисляется выражение-селектор; затем в списках констант ищется такое значение, которое совпадает с полученным значением селектора; далее исполняется оператор, помеченный данной константой. Если такой константы не найдено, то происходит переход к выполнению оператора, следующего после слова Else.

Пример 4. В этом примере демонстрируется использование списка констант в операторе выбора. Программа сообщает, сдал студент экзамен или не сдал. Если оценка одна из следующих: 3, 4, 5, то экзамен сдан; если 2, то не сдан.

```
Case N of
        3,      4,      5:      WriteLn('Экзамен сдан');
                        2:      WriteLn('Экзамен не сдан');
      Else WriteLn('Нет такой оценки')
End
```

Так же как условный оператор, оператор выбора может ис-пользоваться в неполной форме, т. е. без ветви **Else**.

Если применить условный оператор, то эта программа запишется так:

```
If (N=3) or (N=4) or (N=5)
Then WriteLn('Экзамен сдан')
Else If N=2
        Then WriteLn('Экзамен не сдан')
        Else WriteLn('Нет такой оценки');
```

В условии ветвления использовано сложное логическое выражение, содержащее операции логического сложения or (или).

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§18-20, задания №3-5 (§18) и №4 (§19) письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 23
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №10 «Программирование логических выражений».
Тип урока: Урок-практикум.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Практическая работа

Ход работы

Задание

Для каждой задачи составить программу, выводящую значение TRUE*, если указанное высказывание является истинным, и* ***FALSE*** *— в противном случае (использовать условный оператор нельзя).*

1. Треугольник со сторонами **a, b, c** является равносторонним.
2. Целое число **N** является четным двузначным числом.
3. Треугольник со сторонами **a, b, c** является равнобедренным.
4. Среди чисел **a, b, c** есть хотя бы одна пара взаимно противоположных чисел.
5. Данные числа **x, y** являются координатами точки, лежащей в первой координатной четверти.
6. Данные числа **c** и **d** являются соответственно квадратом и кубом числа **a**.
7. Заданное натуральное число **N** является двузначным и кратно **K**.

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§18-20.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 24
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №11 «Программирование ветвящихся алгоритмов».
Тип урока: Урок-практикум.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Практическая работа

Ход работы

Задание
Для каждой задачи составить программу с ветвящейся структурой, используя условный оператор **IF**.

1. Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник. Если да, то прямоугольный ли он.
2. На плоскости ***XOY*** задана своими координатами точка ***A***. Указать, где она расположена: на какой оси или в какой координатной четверти.
3. Грузовой автомобиль выехал из одного города в другой со скоростью υ_1 км/ч. Через Ив этом же направлении выехал легковой автомобиль со скоростью υ_2 км/ч. Составить программу, определяющую, догонит ли легковой автомобиль грузовой через t_1 ч после своего выезда.
4. Написать программу нахождения суммы большего и меньшего из 3 чисел.
5. Написать программу, распознающую по длинам сторон среди всех треугольников прямоугольные. Если таковых нет, то вычислить величину угла ***C***.
6. Найти **max{min(a, b), min(c, d)}**.
7. Составить программу, осуществляющую перевод величин из радианной меры в градусную или наоборот. Программа должна запрашивать, какой перевод нужно осуществить, и выполнять указанное действие.

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§18-20.
Комментарии учителя.
Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 25
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Программирование циклов. Вложенные и итерационные циклы.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Рассмотрим приемы программирования циклов на Паскале. На уроках "Программирование линейных алгоритмов" рассказывалось о том, что существуют две циклические алгоритмические структуры: цикл с предусловием (цикл-пока) и цикл с постусловием (цикл-до). Были показаны способы описания циклических структур в блок-схемах и на Алгоритмическом языке. Форматы соответствующих операторов цикла в Паскале следующие.

Цикл с предусловием (цикл-пока):
While < логическое выражение > **Do**
< оператор >
Цикл с постусловием (цикл-до):
Repeat
< оператор >
Until < логическое выражение >

Различают циклы с заданным числом повторений и итерационные циклы.

На примерах конкретных задач рассмотрим приемы программирования циклов.

В математике известно, что сумма следующего бесконечного числового ряда:

$$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots+\frac{1}{i!}+\ldots$$

в пределе стремится к значению константы e = 2,71828182... Функция ex называется экспонентой, а логарифм по основанию e называется натуральным логарифмом: 1nx.

Требуется составить программу, вычисляющую эту константу по сумме числового ряда. Напомним, что символ «!» читается как «факториал» — функция, определенная следующим образом:

$$x!=\begin{cases}1, & \text{при } x=0;\\ 1\cdot 2\cdot 3\cdot\ldots\cdot x, & \text{при } x>0.\end{cases}$$

Если слагаемые в вычисляемом выражении обозначить так:

$$a_0=1,\ a_1=\frac{1}{1!}=\frac{1}{1},\ a_2=\frac{1}{2!}=\frac{1}{1\cdot 2},\ a_3=\frac{1}{3!}=\frac{1}{1\cdot 2\cdot 3},\ \ldots,$$

то обобщенная формула для i-го элемента будет следующей:

$$a_i = \frac{1}{i!}.$$

Нетрудно увидеть, что между элементами данной последовательности имеется зависимость:

$$a_0 = 1, \quad a_1 = \frac{a_0}{1}, \quad a_2 = \frac{a_1}{2}, \quad a_3 = \frac{a_2}{3} \quad \text{и т. д.}$$

Такая зависимость называется рекуррентной зависимостью, а соответствующая числовая последовательность — рекуррентной последовательностью. Данная рекуррентная последовательность может быть описана следующей ветвящейся формулой, которая называется рекуррентной формулой:

$$a_i = \begin{cases} 1, & \text{при } i = 0; \\ \dfrac{a_{i-1}}{i}, & \text{при } i > 0. \end{cases}$$

Циклы с заданным числом повторений

Пример 1. Дано целое положительное значение ***N***. Требуется вычислить сумму:

$$1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{N!}.$$

Ниже приводятся два варианта программы решения этой задачи. В первом варианте используется цикл с предусловием, во втором — цикл с постусловием.

```
Program Summa_1;
Var E,a: Real; N,i: Integer;
Begin
  Write('N='); ReadLn(N);
  E:=0; i:=0; a:=1;
  While i<=N Do
  Begin
    E:=E+a;
    i:=i+1;
    a:=a/i
  End;
  WriteLn('E=', E)
End.
```

```
Program Summa_2;
Var E,a: Real; N,i: Integer;
Begin
  Write('N='); ReadLn(N);
  E:=0; i:=0; a:=1;
  Repeat
    E:=E+a;
    i:=i+1;
    a:=a/i;
  Until i>N;
  WriteLn('E=', E)
End.
```

Обратите внимание на то, как цикл с предусловием преобразуется в цикл с постусловием — условие цикла помещается после тела цикла и заменяется на противоположное:

Not(i ≤ N) = i > N.

И тот, и другой цикл повторят свое выполнение (N + 1) раз. Переменная i выполняет роль не только знаменателя в дроби 1/i!, но и является счетчиком числа повторений цикла. Такие переменные называются ***параметрами цикла***. И еще: в цикле с постусловием служебные слова ***Repeat и Until*** сами выполняют роль операторных скобок. Поэтому писать ***Begin и End*** здесь не требуется.

Выполнение этих программ на компьютере для значения N = 7 приводит к следующему результату: E=2,7182539.

Для программирования циклов с заданным числом повторений при постоянном шаге изменения параметра цикла в Паскале существует **цикл с параметром**. Вот как выглядит программа решения той же задачи с использованием цикла с параметром:

```
Program Summa_3;
Var E, a: Real; N, i: Integer;
Begin
  Write('N='); ReadLn(N);
  E:=1; a:=1;
  For i:=1 To N Do
  Begin
    a:=a/i;
    E:=E+a
  End;
  WriteLn('E=', E)
End.
```

В программе используется оператор цикла For, для которого существуют два варианта:

1) **For** <параметр цикла>:=<выражение 1> **To** <выражение 2>
Do <оператор>

2) **For** <параметр цикла>:=<выражение 1> **Downto** <выражение 2>
Do <оператор>

Здесь <параметр цикла> — имя простой переменной порядкового типа. Выполнение оператора **For** в первом варианте **(To)** происходит по следующей схеме.

1. Вычисляются значения <выражения 1> и <выражения 2>. Это делается только один раз при входе в цикл.

2. Параметру цикла присваивается значение <выражения 1>.

3. Значение параметра цикла сравнивается со значением <выражения 2>. Если параметр цикла меньше или равен этому значению, то выполняется тело цикла (<оператор>), в противном случае выполнение цикла заканчивается.

4. Значение параметра цикла изменяется на следующее значение в его типе (для целых чисел — увеличивается на единицу); происходит возврат к пункту 3.

Оператор цикла **For** объединяет в себе действия, которые при использовании цикла **While** выполняют различные операторы: присваивание параметру начального значения, сравнение его с конечным значением, изменение значения параметра на следующее.

Во втором варианте оператора **For** слово **Downto** буквально можно перевести как «вниз до». В таком случае параметр цикла изменяется по убыванию, т. е. при каждом повторении цикла параметр изменяет свое значение на предыдущее (равносильно i : =pred (i)).

Работая с оператором **For**, учитывайте следующие правила:

• параметр цикла не может иметь вещественного типа;

• в теле цикла нельзя изменять переменную-параметр цикла;

• при выходе из цикла значение переменной-параметра является неопределенным.

Рассмотрим пример программы, в которой в теле цикла будет присутствовать ветвление.

Пример 2. Составим программу проверки знаний учеником таблицы умножения. Компьютер задает ученику 10 вопросов на умножение чисел от 2 до 9. На каждое задание ученик вводит свой ответ, компьютер сообщает, верный ответ или нет.

На рисунке 3.16 приведена блок-схема такого алгоритма.

Обратите внимание на то, как отображается на блок-схеме цикл с параметром.

В этом алгоритме использована функция random (x), результатом выполнения которой является случайное целое число из диапазона от 0 до x - 1. Следовательно, выражение random (8)+2 принимает случайные значения от 2 до 9. Функция random называется датчиком случайных чисел.

На Паскале этот алгоритм программируется так:

```
Program TablMul;
Begin
  Var x,y,i,z: Integer;
  randomize;
  For i:=1 To 10 Do
  Begin
    x:=random(8)+2;
    y:=random(8)+2;
    WriteLn('Сколько будет ', x, '*', y, '?');
    Read(z);
    If z=x*y
    Then WriteLn('Правильно!')
    Else WriteLn('Неправильно! ', x, '*', y, '=', x*y);
  End;
End.
```

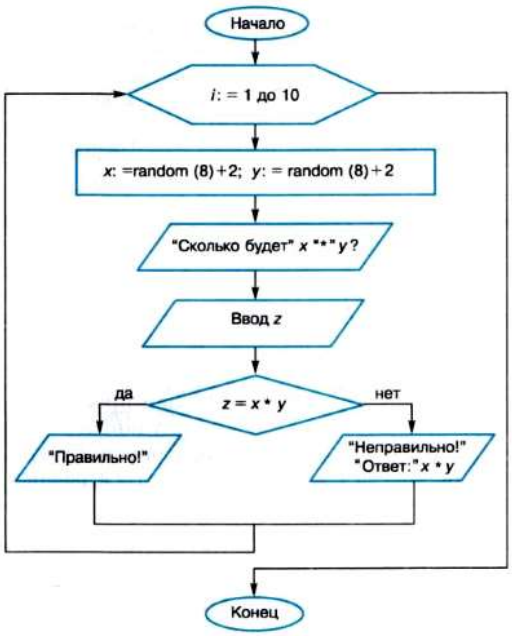


Рис. 3.16. Блок-схема алгоритма из примера 2

А вот фрагмент интерфейса исполнения этой программы:

*Сколько будет 4*8?*
21
*Неправильно! 4*8=32*
*Сколько будет 6*9?*
54
Правильно!

В программе используется стандартная процедура randomize. Ее исполнение производит установку случайного начального состояния датчика случайных чисел. Благодаря этому при повторном выполнении программы будут получаться разные последовательности случайных чисел.

Если в теле одного цикла имеется другой цикл, то такая структура алгоритма называется вложенными циклами. Рассмотрим задачу, программа решения которой имеет структуру вложенных циклов.

Требуется получить на экране компьютера таблицу умножения в форме матрицы Пифагора. Блок-схема алгоритма будет следующей (рис. 3.17).

Здесь цикл по параметру y вложен в цикл по параметру x. Последовательность изменения значений параметров циклов такая:

x = 1; *y* = 1, 2, 3, ..., 9
x = 2; *y* = 1, 2, 3, ..., 9
...
x = 9; *y* = 1, 2, 3, ..., 9

Таким образом, внешний цикл исполнится 9 раз, а внутренний — 9 • 9 = 81 раз. На один шаг повторения внешнего цикла происходит полная прокрутка внутреннего.

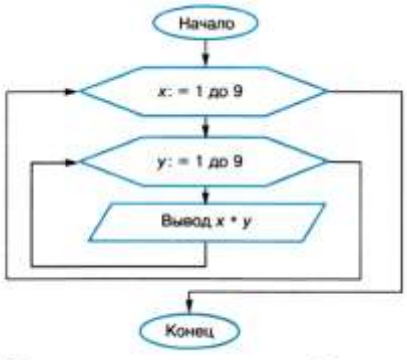


Рис. 3.17. Блок-схема алгоритма получения таблицы умножения

При программировании вложенных циклов используется понятие глубины вложенности. В данном примере глубина вложенности внутреннего цикла равна единице. Если бы внутри вложенного цикла был еще один вложенный цикл (например, для вычисления всех вариантов перемножения трех сомножителей), то его глубина вложенности равнялась бы двум.

Программа на Паскале получения матрицы Пифагора:

```
Program MatrPif;
Var x, y: 1..9;
Begin
  For x:=1 To 9 Do
  Begin
    WriteLn;
    For y:=1 To 9 Do
      Write(x*y:3)
  End
End.
```

В результате выполнения программы на экране получим:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

В программе присутствуют некоторые элементы, не отраженные в блок-схеме. В описании переменных X, Y использован ограниченный тип: 1. . 9, поскольку в данной задаче величины принимают целые значения только в этом диапазоне. Оператор ***WriteLn*** перед началом вложенного цикла обеспечивает переход к новой строке в таблице каждый раз при смене первого сомножителя. В операторе ***Write(X*Y:3)*** для вывода значения произведения после двоеточия использован указатель формата — 3. Это обеспечивает вывод чисел в три позиции на экране, благодаря чему соответствующие столбцы таблицы располагаются строго друг под другом. Первая строка и первый столбец на экране — это сомножители, что соответствует стандартному формату таблицы Пифагора.

Итерационные циклы

Итерационный цикл — это цикл, для которого число повторений тела цикла заранее неизвестно. В итерационных циклах на каждом шаге вычислений происходят последовательное приближение и проверка условия достижения искомого результата. Выход из итерационного цикла осуществляется в случае выполнения заданного условия.

Пример 1. Снова рассмотрим задачу вычисления суммы числового ряда:

$$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots+\frac{1}{i!}+\ldots$$

Но теперь условие будет таким: в сумму нужно включить только слагаемые, значение которых больше некоторой малой величины **ε**. При этом полученная сумма будет отличаться от предельного значения (константы **e**) на величину, не большую **ε**.

Поскольку с увеличением значения **i** величина **1/i!** уменьшается, в сумму надо включать все слагаемые, предшествующие первому значению, меньшему **ε**. Вот две программы решения этой задачи, использующие циклы с предусловием и постусловием:

```
Program Summa_4;
Var E,a,eps: Real; i:Integer;
Begin
  Write('Eps='); ReadLn(eps);
  E:=0; i:=0; a:=1;
  While a>eps Do
  Begin
    E:=E+a;
    i:=i+1;
    a:=a/i
  End;
  WriteLn('E=', E,
          'Слагаемых:', i)
End.
```

```
Program Summa_5;
Var E,a,eps: Real; i:Integer;
Begin
  Write('Eps='); ReadLn(eps);
  E:=0; i:=0; a:=1;
  Repeat
    E:=E+a;
    i:=i+1;
    a:=a/i;
  Until a<=eps;
  WriteLn('E=', E,
          'Слагаемых:', i)
End.
```

Решить эту задачу, используя цикл с параметром, нельзя. Итерационные циклы программируются путем использования либо **цикла-пока**, либо **цикла-до**.

В качестве результата выводится значение суммы и число вошедших в нее слагаемых. Выполнение этих программ для значения $\varepsilon=10^{-8}$ дает в результате: *E=2,71828182, Слагаемых: 12*. Таким образом, за 12 повторений цикла значение константы e получено с точностью до 8 знаков после запятой. Слово «итерации» означает «приближения». С каждым повторением цикла вычисляемая величина приближалась к предельному значению константы.

Пример 2. На уроках "Программирование линейных алгоритмов" была рассмотрена задача вычисления суммы цифр трехзначного натурального числа. Программа имела линейную структуру. Поставим задачу в более общем виде: для любого многозначного натурального числа вычислить сумму всех его цифр.

Выделение цифр происходит с помощью однотипных действий: использования операций **mod** и **div**. Очевидно, что их можно «зациклить». Однако число повторений цикла будет разным для чисел разной длины. Поэтому эта задача не решается с помощью цикла с заданным числом

повторений. В таком случае в программе можно использовать либо оператор цикла **While**, либо **Repeat** и нельзя — цикл с параметром **For**.

Программа с использованием цикла с предусловием:

```
Program SumCifr;
var X: Longint; Sum: Word;
Begin
  Write('Введите целое число: '); ReadLn(X);
  Sum:=0;
  While (X>0) Do
  Begin
    Sum:=Sum + X mod 10; //прибавление к сумме
                         //младшей цифры
    X:=X div 10          //отбрасывание в X
                         //младшей цифры
  End;
  WriteLn('Сумма цифр = ', Sum)
End.
```

Поскольку при каждом повторении цикла от числа **X** отбрасывается одна младшая цифра, закончить цикл нужно тогда, когда **X** станет равным нулю. Обратите внимание на типы переменных. Надо помнить о разнообразии групп типов в Паскале. Назначение переменной **X** типа **Longint** дает возможность вводить в нее значения, включающие до десяти знаков. Для переменной **Sum**, назначен тип **Word**, поскольку сумма цифр может быть только положительным числом.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §§21,22, задания №1,2 (§21) и №3 (§22) письменно в тетради. Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 26
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №12 «Программирование циклических алгоритмов».
Тип урока: Урок-практикум.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Практическая работа

Ход работы

Задание
Циклы с заданным числом повторений. Вычислить значение суммы или произведения числовой последовательности.

1. Дано натуральное число ***N***. Вычислить:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{N}.$$

2. Дано натуральное число ***N***. Вычислить:

$$S = 1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \ldots + \frac{1}{2N+1}.$$

3. Дано натуральное число ***N***. Вычислить:

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \ldots + (-1)^N \cdot \frac{1}{2^N}.$$

4. Дано натуральное число ***N***. Вычислить:

$$S = \frac{1}{\sin 1} + \frac{1}{\sin 1 + \sin 2} + \ldots + \frac{1}{\sin 1 + \sin 2 + \ldots + \sin N}.$$

5. Дано натуральное число ***N***. Вычислить произведение первых ***N*** сомножителей:

$$P = \frac{2}{3} \cdot \frac{4}{5} \cdot \frac{6}{7} \cdot \ldots \cdot \frac{2N}{2N+1}.$$

6. Дано натуральное ***n***. Вычислить:

$$\frac{2}{1}+\frac{3}{2}+\frac{4}{3}+\ldots+\frac{n+1}{n}.$$

7. Вычислить:

$$(1+\sin 0{,}1)(1+\sin 0{,}2)\cdot\ldots\cdot(1+\sin 10).$$

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§21,22.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 27
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Вспомогательные алгоритмы и подпрограммы.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Еще одним важнейшим методологическим приемом структурного программирования является ***декомпозиция решаемой задачи на подзадачи*** — более простые, с точки зрения программирования, части исходной задачи. Алгоритмы решения таких подзадач называются **вспомогательными алгоритмами**.

В языках программирования вспомогательные алгоритмы называются подпрограммами. В Паскале различаются две разновидности подпрограмм: процедуры и функции. Рассмотрим этот вопрос на примере следующей задачи: даны два натуральных числа **a** и **b**. Требуется определить наибольший общий делитель трех величин: $a + b$, $a^2 + b^2$, $a \cdot b$.

Запишем это так: НОД($a + b$, $a^2 + b^2$, $a \cdot b$).

Идея решения состоит в следующем математическом факте: если **x, y, z** — три натуральных числа, то НОД(**x, y, z**) = НОД(НОД(**x, y**), **z**). Иначе говоря, нужно найти НОД двух величин, а затем НОД полученного значения и третьего числа (попробуйте это доказать).

Очевидно, что вспомогательным алгоритмом для решения поставленной задачи является алгоритм получения наибольшего общего делителя двух чисел. Эта задача решается с помощью алгоритма Евклида, который подробно обсуждался в 9 классе. Напомним, что идея алгоритма Евклида основана на следующей формуле:

$$\text{НОД}(M,N)=\begin{cases} M, \text{ при } M=N; \\ \text{НОД}(M-N,\ N), \text{ при } M>N; \\ \text{НОД}(N,\ N-M), \text{ при } N>M. \end{cases}$$

```
процедура Евклид(цел M, N, K)
нач
  пока M<>N
  нц
     если M>N
     то M:=M–N
     иначе N:=N–M
     все
  кц
  K:=M
кон

алг задача
цел a, b, c
нач
  ввод(a, b)
  Евклид(a+b, a*a+b*b, c)
  Евклид(c, a*b, c)
  вывод(c)
кон
```

Приведем алгоритм решения поставленной задачи на учебном Алгоритмическом языке. Алгоритм состоит из процедуры «Евклид» и основного алгоритма «Задача», в котором присутствуют два обращения к процедуре:

Здесь **M, N** и **K** являются формальными параметрами процедуры. **M** и **N** — параметры-аргументы, **K** — параметр-результат.

Процедуры в Паскале. Основное отличие процедур в Паскале от процедур в Алгоритмическом языке (АЯ) состоит в том, что процедуры в Паскале описываются в разделе описания

подпрограмм, а в АЯ процедура является внешней по отношению к вызывающей программе. Теперь посмотрим, как решение поставленной задачи программируется на Паскале.

```
Program NOD1;
Var A, B, C: Integer;
Procedure Evklid(M, N: Integer; Var K: Integer);
Begin
  While M<>N Do
    If M>N
    Then M:=M-N
    Else N:=N-M;
  K:=M
End;
Begin
  Write('a = '); ReadLn(A);
  Write('b = '); ReadLn(B);
  Evklid(A+B, A*A+B*B), C);
  Evklid(C, A*B, C);
  WriteLn('НОД = ', C)
End.
```

В данном примере **обмен** аргументами и результатами между основной программой и процедурой производится **через параметры**. Описание процедуры на Паскале имеет следующий формат:

Procedure <имя процедуры> [(список формальных параметров)]; <блок>

Квадратные скобки указывают на то, что список формальных параметров может отсутствовать, т. е. возможна процедура без параметров. Параметры могут быть параметрами-переменными и параметрами-значениями. Параметры-переменные записываются следующим образом:

Var <список переменных>: <тип>

Параметры-значения указываются так:

<список переменных>: <тип>

Чаще всего аргументы представляются как параметры-значения (хотя могут быть и параметрами-переменными). А для передачи результатов используются параметры-переменные. Процедура в качестве результата может передавать в вызывающую программу множество значений (в частном случае — одно), а может и ни одного. Теперь рассмотрим правила обращения к процедуре. Обращение к процедуре производится в форме **оператора процедуры**:

<имя процедуры>[(список фактических параметров)]

Если описана процедура с формальными параметрами, то обращение к ней производится оператором процедуры с фактическими параметрами. Правила соответствия между формальными и фактическими параметрами: соответствие ***по количеству***, соответствие ***по последовательности*** и соответствие ***по типам***.

Взаимодействие формальных и фактических параметров через параметры-переменные называется передачей по ссылке: при обращении к процедуре ей передается ссылка на переменную, заданную в качестве фактического параметра. Эта ссылка и используется процедурой для доступа к этой переменной.

Другой вариант взаимодействия формальных и фактических параметров называется передачей по значению: вычисляется значение фактического параметра (выражения), и это значение присваивается соответствующему формальному параметру.

В рассмотренном нами примере формальные параметры **M** и **N** являются параметрами-значениями. Это аргументы процедуры. При обращении к ней первый раз им соответствуют значения выражений **A + B** и **abs(A - B)**; второй раз — **C** и **A*B**. Параметр **K** является параметром-переменной. В ней получается результат работы процедуры. В обоих обращениях к процедуре соответствующим фактическим параметром является переменная **C**. Через эту переменную основная программа получает результат.

```
Program NOD2;
Var A, B, K, M, N: Integer;
Procedure Evklid;
Begin
  While M<>N Do
    If M>N
    Then M:=M-N
    Else N:=N-M;
  K:=M
End;
Begin
  Write('a= '); ReadLn(A);
  Write('b= '); ReadLn(B);
  M:=A+B; N:=A*A+B*B;
  Evklid;
  M:=K; N:=A*B;
  Evklid;
  WriteLn('НОД равен ', K)
End.
```

Теперь рассмотрим другой вариант программы, решающей ту же задачу. В ней используется процедура без параметров.

Чтобы разобраться в этом примере, требуется объяснить новое для нас понятие: *область действия описания*.

Областью действия описания любого программного объекта (переменной, типа, константы и т. д.) является тот блок, на который это описание распространяется. Если данный блок вложен в другой (подпрограмма), то присутствующие во вложенном блоке описания являются **локальными**. Они действуют только в пределах внутреннего блока. Описания же, расположенные во внешнем блоке, называются **глобальными** по отношению к внутреннему блоку. Если глобально описанный объект используется во внутреннем блоке, то на него распространяется внешнее (глобальное) описание.

В программе **NOD1** переменные **M, N, K** являются локальными внутри процедуры; переменные **A, B, C** — глобальные. Однако внутри процедуры

переменные **A, B, C** не используются. Связь между внешним блоком и процедурой осуществляется через параметры.

В программе **N0D2** все переменные являются глобальными. В процедуре **Evklid** нет ни одной локальной переменной (нет и параметров). Переменные **M** и **N**, используемые в процедуре, получают свои значения через оператор присваивания в основном блоке программы и изменяют значения в подпрограмме. Результат получается в глобальной переменной **K**, значение которой выводится на экран. Здесь **обмен** значениями между основной программой и процедурой производится **через глобальные переменные**.

Использование механизма передачи через параметры делает процедуру более универсальной, независимой от основной программы. Однако в некоторых случаях оказывается удобнее использовать передачу через глобальные переменные. Чаще такое бывает с процедурами, работающими с большими объемами информации. В этой ситуации глобальное взаимодействие экономит память компьютера.

Функции. Теперь выясним, что такое подпрограмма - функция. Обычно функция используется в том случае, когда результатом работы подпрограммы должна быть скалярная (простая) величина. Тип результата называется **типом функции**. Формат описания функции следующий:

```
Function <имя функции> [(<список формальных
                         параметров>)]: <тип функции>;
  <блок>
```

Как и у процедуры, у функции в списке формальных параметров могут присутствовать параметры-переменные и параметры-значения. Всё это — **аргументы функции**. Параметры вообще могут отсутствовать, если аргументы передаются глобально.

Программа решения рассмотренной выше задачи с использованием функции будет выглядеть следующим образом:

```
Program NOD3;
Var A, B, Rez: Integer;
Function Evklid(M, N: Integer): Integer;
Begin
  While M<>N Do
    If M>N
    Then M:=M-N
    Else N:=N-M;
  Evklid:=M
End;
Begin
  Write('a = '); ReadLn(A);
  Write('b = '); ReadLn(B);
  Rez:=Evklid(Evklid(A+B, A*A+B*B), A*B);
  WriteLn('NOD равен ', Rez)
End.
```

Из примера видно, что тело функции отличается от тела процедуры только тем, что в функции ***результат присваивается идентификатору функции***: *Evklid:=M*.

Обращение к функции является операндом в выражении. Оно записывается в следующей форме:

<имя функции> (<список фактических параметров>)

Правила соответствия между формальными и фактическими параметрами все те же. Сравнивая приведенные выше программы, можно сделать вывод, что программа N0D3 имеет определенные преимущества перед другими. Функция позволяет получить результат путем выполнения одного оператора присваивания. Здесь также иллюстрируется возможность того, что фактическим аргументом при обращении к функции может быть эта же функция.

По правилам стандарта Паскаля, возврат в вызывающую программу из подпрограммы происходит, когда выполнение подпрограммы доходит до ее конца (последний **End**). Однако в современных версиях Паскаля есть средство, позволяющее выйти из подпрограммы в любом ее месте. Это оператор-процедура **Exit**. Например, функцию определения большего из двух данных вещественных чисел можно описать так:

```
Function Max(X, Y: Real): Real;
Begin
  Max:=X;
  If X>Y Then Exit Else Max:=Y
End;
```

Модифицированный алгоритм Евклида. Подпрограмму алгоритма Евклида можно составить иначе, если воспользоваться операцией **mod** (получение остатка от деления), имеющейся в Паскале. Идея алгоритма исходит из справедливости следующих равенств:

$$\text{НОД}(M,N)=\begin{cases} M, \text{ при } N=0; \\ \text{НОД}(N,\ M \bmod N), \text{ при } N\neq 0. \end{cases}$$

В таком случае функцию Evklid можно переписать так:

```
Function Evklid(M, N: Integer): Integer;
Var R: Integer;
Begin
  While N<>0 Do
    Begin R:=M mod N; M:=N; N:=R End;
  Evklid:=M
End;
```

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §23, задание №4 письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 28
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №13 «Программирование с использованием подпрограмм».
Тип урока: Урок-практикум.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Практическая работа

Ход работы

Задание
Для решения всех задач сделать два варианта программы: с реализацией указанной подпрограммы в виде функции и в виде процедуры.

1. Составить программу нахождения наибольшего общего делителя (НОД) и наименьшего общего кратного (НОК) двух натуральных чисел

$$\text{НОК}(A,B)=\frac{A\cdot B}{\text{НОД}(A,B)}.$$

Использовать подпрограмму алгоритма Евклида для определения НОД.

2. Вычислить площадь правильного шестиугольника со стороной **a**, используя подпрограмму вычисления площади треугольника.

3. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ — (A, B, C, D — натуральные числа).
Составить программу деления дроби на дробь. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.

4. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ — (A, B, C, D — натуральные числа).
Составить программу умножения дроби на дробь. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.

5. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ — (A, B, C, D — натуральные числа).
Составить программу вычитания из первой дроби второй. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.

6. Написать программу вычисления суммы $1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}$ — для заданного числа **n**. Результат представить в виде несократимой дроби $\frac{p}{q}$ (**p, q** — натуральные). Использовать подпрограммы алгоритма Евклида для определения НОД и сложения двух простых дробей.

7. Даны числа **X, Y, Z, T** — длины сторон четырехугольника. Вычислить его площадь, если угол между сторонами длиной **X** и **Y** — прямой. Использовать две подпрограммы для вычисления площадей: прямоугольного треугольника и прямоугольника.

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §23.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 29
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Массивы.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Массивом в Паскале называют переменную величину регулярного типа.

Регулярный тип — это структурный тип данных, представляющих собой совокупность пронумерованных однотипных величин.

Описание массивов. Переменная регулярного типа описывается в разделе описания переменных в следующей форме:

Var <идентификатор>: **array**[<тип индекса>] **of** <тип компонентов>

В данном случае **квадратные скобки** — это обязательные символы, которые называются индексными скобками. Чаще всего в качестве типа индекса употребляется ограниченный тип. Например, массив вещественных чисел, хранящий 12 значений среднемесячных температур в течение года, опишется так:

Var T: **array**[1..12] **of** Real;

Описание массива определяет, во-первых, размещение массива в памяти, во-вторых, правила его дальнейшего употребления в программе.

Элемент массива идентифицируется в виде переменной с индексами:

<Идентификатор массива> [<Индексы элемента>]

Для одномерного массива индекс — это одно значение. Для многомерных массивов индекс — множество значений. В качестве индекса может употребляться любое выражение соответствующего типа. Например, для элементов массива температур возможны обозначения: T[5] , T[k], T[i + j], T[m div 2].

Последовательные элементы массива располагаются в последовательных ячейках памяти (T [1], T[2] и т. д.), причем значения индекса не должны выходить за диапазон 1. .12.

Тип индекса может быть любым скалярным порядковым типом, кроме ***Integer***. Например, в программе могут присутствовать следующие описания:

Var cod: **array**[Char] of 1..100; L: **array**[Boolean] **of** Char;

В такой программе допустимы следующие обозначения элементов массивов:

cod['x']; Lftrue]; cod[chr(65)] ; L[a>0].

В некоторых случаях бывает удобно в качестве индекса использовать перечислимый тип. Например, данные о количестве учеников в четырех десятых классах одной школы могут храниться в следующем массиве:

Type Index = (A, B, C, D) ;

Var class_10: **array**[Index] **of** Byte;

И если, например, элемент class 10 [А] равен 35, то это означает, что в 10А классе 35 человек. Такое индексирование улучшает наглядность программы.

Часто структурному типу присваивается имя в разделе типов, которое затем используется в разделе описания переменных.

Type Masl = **array** [1..100] **of** Integer;

Mas2 = **array** [-10.. 10] **of** Char;

Var Num: Masl; Sim: Mas2;

До сих пор речь шла об одномерных массивах, в которых типы элементов скалярные.

Многомерный массив в Паскале трактуется как одномерный массив, тип элементов которого также является массивом (массив массивов).

В качестве примера рассмотрим таблицу с информацией о среднемесячных температурах за 10 лет, например с 2001 по 2010 год. Очевидно, для этого удобна прямоугольная (двумерная) таблица, в которой столбцы соответствуют месяцам, а строки — годам.

Год	Месяц											
	1	2	3	4	5	6	7	8	9	10	11	12
2001	–23	– 17	– 8,4	6,5	14	18,6	25	19	12.3	5,6	– 4,5	– 19
2002	– 16	– 8,5	–3,2	7,1	8,4	13,8	28,5	21	6,5	2	– 13	– 20
2003	– 9,8	– 14	–9,2	4,6	15,6	21	17,8	20	11,2	8,1	– 16	– 21
...	...	...	...	...	...	...	...	...	...	...	...	...
2010	– 25	– 9,7	– 3,8	8,5	13,9	17,8	23,5	17,5	10	5,7	– 14	– 20

Для обработки такой таблицы в программе следует описать массив:

Var Tabl: **array**[2001..2010] **of array**[1.. 12] **of** Real;

Вот примеры обозначения некоторых элементов этого массива:

Tabl [2001] [1] ; Tabl [2005] [10] ; Tabl [2010] [12] .

Однако чаще употребляется другая, эквивалентная форма обозначения элементов двумерного массива:

Tabl [2 001, 1] ; Tabl [2005, 10] ; Tabl [2010,12] .

Переменная TaЫ [2 001] обозначает всю первую строку таблицы, т. е. весь массив температур за 2001 год. Другим эквивалентным вариантом приведенному выше описанию является следующее:

Type Month = **array** [1..12] **of** Real;

Year = **array** [2001..2010] **of** Month;

Var Tabl: Year;

Наиболее краткий вариант описания данного массива такой:

Var Tabl: **array** [2001.. 2010, 1..12] **of** Real;

Продолжая по аналогии, можно определить трехмерный массив как одномерный массив, у которого элементами являются двумерные массивы. Вот пример описания трехмерного массива:

Var A: **array**[1..10, 1..20, 1..30] **of** Integer;

Это массив, состоящий из 10 • 20 • 30 = 6000 целых чисел и занимающий в памяти 6000 • 2 = 12 000 байтов. В Паскале нет ограничения сверху на размерность массива. Однако в каждой конкретной реализации Паскаля ограничивается объем памяти, выделяемый под массивы. В Турбо Паскале это ограничение равно 64 килобайтам.

По аналогии с математикой одномерные числовые массивы часто называют векторами, а двумерные — матрицами.

В Паскале не допускается употребление динамических массивов, т. е. таких, размер которых определяется в процессе выполнения. Изменение размеров массива происходит через изменение в тексте программы и повторную компиляцию. Для упрощения таких изменений удобно определять индексные параметры в разделе констант:

Const Imax = 10; Jmax = 20;

Var Mas: **array**[1..Imax, l..Jmax] **of** Integer;

Теперь для изменения размеров массива Mas и всех операторов программы, связанных с этими размерами, достаточно отредактировать только одну строку в программе — раздел констант.

Действия над массивом как единым целым. Такие действия допустимы лишь в двух случаях:

- присваивание значений одного массива другому;
- применение к массивам операций отношения «равно», «не равно».

В обоих случаях массивы должны иметь одинаковые типы (тип индексов и тип элементов).

Пример 1

Var P, Q: **array** [1.. 5, 1..10] **of** Real;

При выполнении операции присваивания

P:=Q

все элементы массива **P** станут равными соответствующим элементам массива **Q**.

Пример 2

Как уже отмечалось, в многомерных массивах переменная с индексом может обозначать целый массив. Тогда если массив TaЬI описан так:

Type mas = **array** [1..12] **of** Real;

Var Tabl: **array** [2001.. 2010] **of** mas;

и в нем требуется данные за 2009 год сделать такими же, как за 2001 год (девятой строке присвоить значение первой строки), то это можно сделать одним присваиванием:

Tabl [2009] : =Tabl [2001]

А если нужно поменять местами значения этих строк, то это делается через третью переменную того же типа:

P:=Tabl [2009] ; Tabl [2009] : =Tabl [2001] ; Tabl [2001] :=P; где **P** описана так:

Var P: mas;

Ввод и вывод массивов производятся покомпонентно. Вот примеры ввода с клавиатуры значений одномерного и двумерного массивов:

For I: =1 **To** 12 **Do**

ReadLn(T[I]);

For I:=l To Imax Do

For J:=l **To** Jmax **Do**

ReadLn(Mas[I, J]);

Здесь каждое следующее значение будет вводиться с новой строки. Для построчного ввода используется оператор **Read**.

Аналогично в цикле по индексной переменной организуется вывод значений массива на экран. Например:

For I: =1 **To** 12 **Do** Write (T [I] : 8 : 4) ;

Напомним, что модификатор формата 8:4 означает вывод числа в формате с фиксированной точкой в 8 позициях, из которых в 4 последних позициях размещается дробная часть.

Следующий фрагмент программы организует построчный вывод матрицы на экран:

```
For I:=1 To Imax Do
Begin
  For J:=1 To Jmax Do
    Write(Mas[I,J]:6);
  WriteLn
End;
```

После вывода очередной строки матрицы оператор **Writeln** без параметров переведет курсор в начало новой строки. Следует заметить, что в последнем примере матрица на экране будет получена в естественной форме прямоугольной таблицы, если Jmax не превышает 12 (подумайте почему).

Заполнение массива

Значения массива могут задаваться вводом с клавиатуры, чтением из файла или вычислением в программе. В некоторых задачах статистического характера требуется заполнять массивы случайными числами.

Пример 1. Заполнить массив равномерно распределенными целыми случайными числами в диапазоне от 0 до 100.

```
Var X: array[1..20] of Integer; i: Integer;
Begin
  Randomize;
  For i:=1 To 20 Do X[i]:=Random(100);
  For i:=1 To 20 Do Write(X[i]:4)
End.
```

Со стандартной функцией **Random (x)** вы уже знакомы. Напомним, что она возвращает псевдослучайное целое число в диапазоне **от 0 до x - 1**.

Если требуется изменить диапазон случайных чисел, то это всегда можно сделать путем сдвига. Например, если нужно получить числа в диапазоне от -50 до 50, то в программе пишется оператор присваивания:

X[i]:=Random(100)-50;

Для получения вещественных случайных чисел используется функция **Random** без аргумента. Она возвращает случайные дробные значения в диапазоне [0,1). С помощью сдвига и множителя эти значения можно привести к любому диапазону. Например, следующее выражение будет вычислять случайное вещественное число в диапазоне значений от -5 до 5: 10 * Random-5.

Пример 2. Заполнить верхнетреугольную матрицу указанного вида и вывести ее на экран.

Пояснение: для элементов **M[i, j]** матрицы M, расположенных в верхнем треугольнике (включая диагональ), выполняется следующее соотношение между индексами: **j ≤ i.**

Матрица:

```
1 2 3 4
0 2 3 4
0 0 3 4
0 0 0 4
```

Программа решения задачи:

```
Var M: array[1..4,1..4] of Integer;
    i, j: Integer;
Begin
  For i:=1 To 4 Do
    For j:=1 To 4 Do
      If j<i Then M[i,j]:=0 Else M[i,j]:=j;
  For i:=1 To 4 Do
    Begin
      For j:=1 To 4 Do
        Write (M[i,j]:3);
      Writeln
    End
End.
```

Пример 3. Выбор максимального элемента. В одномерном массиве **X** из примера 1 требуется определить наибольшее значение среди значений элементов и его порядковый номер (индекс).

Идея алгоритма решения этой задачи следующая: чтобы в переменной **XMax** получить максимальное значение массива **X**, сначала в нее заносится первое значение массива **X[1]**. Затем значение **XMax** поочередно сравнивается с остальными элементами массива, и каждое значение, большее **Xmax**, присваивается этой переменной. Для получения номера максимального элемента массива в целочисленной переменной **imax** следует записывать в нее номер элемента массива X одновременно с занесением значения в **Xmax**. На Алгоритмическом языке это запишется так:

```
Xmax:=X[1]; imax:=1
для I:=2 до 20
нц
  если X[I]>Xmax
  то
    Xmax:= X[I]; imax:=I
  все
кц
```

Заметим, что если в массиве **X** несколько значений, равных максимальному, то в **imax** будет получен первый номер из этих элементов. Чтобы получить номер последнего элемента, равного максимальному, нужно в ветвлении если заменить знак отношения > на >=. Для нахождение минимального элемента массива достаточно заменить знак отношения «больше» на «меньше».

Оформим в виде процедуры на Паскале подпрограмму поиска максимального элемента в одномерном массиве. Заполним одномерный массив случайными числами (как в примере 1). С помощью процедуры найдем в нем максимальное значение и индекс его первого вхождения в массив.

Процедура MaxArray имеет три параметра: исходный массив **A, Max A** — переменную для найденного максимального значения, **k** — переменную для индекса максимального значения. При обращении к процедуре им соответствуют фактические параметры: **X, Xmax, imax.** Размер массива определяется глобальной константой **n**, значение которой используется как в основной программе, так и в процедуре.

```
Const n = 20;
Type mas = array[1..n] of Integer;
Var X: mas; i, Xmax, imax: Integer;
{Начало описания процедуры}
Procedure MaxArray(Var A:mas; Var MaxA, k:Integer);
Var j: Integer;
Begin MaxA:=A[1]; k:=1;
  For j:=2 To n Do
    If A[j]>MaxA Then
      Begin MaxA:=A[j]; k:=j End
End; {Конец описания процедуры}
Begin
  Randomize;
  For i:=1 To n Do X[i]:=Random(100); {Заполнение
                                       массива}
  MaxArray(X, Xmax, imax); {Обращение к процедуре}
  Write('Xmax=', Xmax, 'imax=', imax) {Вывод
                                       результатов}
End.
```

Пример 4. Сортировка массива. В одномерном массиве **X из N** элементов требуется произвести перестановку значений так, чтобы они расположились по возрастанию, т. е. $X_1 \le X_2 \le ... \le X_N$.

Существует целый класс алгоритмов сортировки. Ниже описан алгоритм, который называется методом пузырька.

Идея алгоритма: производится последовательное упорядочивание смежных пар элементов массива: X_1 и X_2, X_2 и X_3,..., X_{N-1} и X_N. В итоге максимальное значение переместится в X_N. Затем ту же процедуру повторяют до X_{N-1} и т. д., вплоть до цепочки из двух элементов X_1 и X_2. Такой алгоритм будет иметь структуру двух вложенных циклов, причем внутренний цикл переменной (сокращающейся) длины.

```
для I:=1 до N-1
нц
  для J:=1 до N-I
  нц
    если X[J]>X[J+1]
    то
      A:=X[J];
      X[J]:=X[J+1];
      X[J+1]:=A
    все
  кц
кц
```

Для сортировки массива по убыванию значений достаточно заменить знак отношения «больше» на «меньше».

Запрограммируем на Паскале процедуру сортировки массива по возрастанию методом пузырька.

```
Const n=20;
Type: vector = array[1..n] of Real;
{описание процедуры сортировки}
Procedure SortArray(Var X: vektor);
Var i,j: Integer; A: Real;
Begin
  For i:=1 To n-1 Do
    For j:=1 To n-i Do
      If X[j]>X[j+1] Then
        Begin A:=X[j]; X[j]:=X[j+1]; X[j+1]:=A End
End; {Конец описания процедуры}
```

Пример 5. Ранее было рассмотрено описание двумерного массива, содержащего среднемесячные температуры за 10 лет, с 2001 по 2010 год. Определить, в каком году за этот период было самое теплое лето, т. е. в каком году была наибольшая средняя температура летних месяцев.

Идея решения: в одномерном массиве **S** получить средние температуры летних месяцев за каждый год из 10 лет. Затем найти номер наибольшего элемента в этом массиве, это и будет искомый год.

```
Program Example_2;
Type Month = array[1..12] of Real;
     Year = array[2001..2010] of Month;
Var Tabl: Year;
    S: array[2001..2010] of Real;
    I, J, K: Integer;
Begin {Ввод данных с клавиатуры}
  For I:=2001 To 2010 Do
    For J:=1 To 12 Do
    Begin
      Write(J:2, '.', I:4, ': ');
      ReadLn(Tabl[I, J])
    End;
  {Вычисление вектора средних летних температур}
  For I:=2001 To 2010 Do
  Begin
    S[I]:=0;
    For J:=6 To 8 Do
      S[I]:=S[I]+Tabl[I, J];
    S[I]:=S[I]/3
  End;
  {Определение года с самым теплым летом}
  K:=2001;
  For I:=2002 To 2010 Do
    If S[I]>S[K] Then K:=I;
  WriteLn('Самое теплое лето было в ',
          K, '-м году')
End.
```

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §24, вопросы к параграфу устно.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 30
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №14 «Программирование обработки одномерных и двумерных массивов».
Тип урока: Урок-практикум.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Практическая работа

Ход работы

Задание 1
Составить программу решения поставленной задачи по обработке одномерного массива (вектора). По возможности, использовать подпрограммы.

1. Дана последовательность действительных чисел a_1, a_2, …, a_n. Выяснить, будет ли она возрастающей.
2. Дан массив из **N** действительных чисел. Подсчитать, сколько в нем отрицательных, положительных и нулевых элементов.
3. Даны действительные числа a_1, a_2, …, a_n. Поменять местами первый наибольший элемент с последним наименьшим элементом.
4. В заданном одномерном массиве поменять местами соседние элементы, стоящие на четных местах, с элементами, стоящими на нечетных местах.
5. Задана последовательность $\{X_i\}$ из **N** вещественных чисел. Вычислить последовательность $\{S_i\}$ по формуле:

$$S_i = \sqrt{\frac{(X_i - M)^2}{N-1}},$$

где **M** — среднее арифметическое значение последовательности **X**.

Задание 2
Составить программу решения поставленной задачи по обработке двумерного массива (матрицы). По возможности, использовать подпрограммы.

1. Вычислить сумму и число положительных элементов матрицы **A[N, N]**, находящихся над главной диагональю.
2. Дана целая квадратная матрица **n-го** порядка. Определить, является ли она магическим квадратом, т. е. такой матрицей, в которой суммы элементов во всех строках и столбцах одинаковы.

3. Определить, является ли заданная целая квадратная матрица **n-го** порядка симметричной (относительно главной диагонали).

4. Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали в этой же строке.

5. Упорядочить по возрастанию элементы каждой строки матрицы размером n x m.

3. Подведение итогов урока

Оценки за урок, следующие _____

4. Домашнее задание

Записываем домашнее задание: §24.

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 31
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Символьный тип данных. Строки символов.
Тип урока: Комбинированный урок.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Объяснение нового материала (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Объяснение нового материала

Символьный тип данных

Величина типа **«символ»** может принимать значения любых символов компьютерного алфавита. Символьная величина занимает 1 байт памяти, в котором хранится код этого символа, соответствующий используемой кодовой таблице. Заметим, что в Delphi наряду с однобайтовой кодировкой символов используется и двухбайтовая.

Символьная константа записывается между апострофами. Например: 1R', ' + ', '9', 'j'.

Символьной тип называется Char. Пример описания символьных переменных:

Var c1, c2: Char;

Символьный тип относится к порядковым типам данных. Из этого следует:

• символы — упорядоченное множество;
• у каждого символа в этом множестве есть свой порядковый номер;
• между символами работает соотношение «следующий — предыдущий ».

Порядковый номер символа — это его десятичный код, который лежит в диапазоне от 0 до 255. Например, в кодовой таблице ASCII десятичный код латинской буквы 'A' равен 65, а цифры '5' — 53. О стандартах кодирования символов рассказывалось на уроках 13 - 16 "Представление текста, изображения и звука в компьютере".

Функция Ord(x)

Ord(x) — функция от аргумента порядкового типа, которая возвращает порядковый номер значения **x** в этом типе данных. Если **x** — символьная величина, то результатом функции будет десятичный код **x** в кодовой таблице. Например:

Ord('A')= 65, Ord('5')= 53

Функция Chr (x)

Chr (x) — функция от целочисленного аргумента, результатом которой является символ с кодом, равным **x**. Например:

Chr(65)='A', Chr(53)='5'

Поскольку коды символов лежат в диапазоне от 0 до 255, желательно тип **x** определять либо как **byte**, либо как интервальный тип 0..255.

Пример 1. Составить программу на Паскале, по которой на экран будет выводиться таблица кодировки в диапазоне кодов от 32 до 255. Напомним, что символы с кодами, меньшими 32, являются управляющими (не экранными).

```
Program Tabl_code;
Var kod: Byte; {Целые числа от 0 до 255}
Begin
  For kod:=32 To 255 Do {Перебор кодов символов}
  Begin
    If (kod mod 10=0) Then Writeln;   {Перевод строки
                                      через 10 шагов}
    Write(chr(kod):3, kod:4);         {Вывод символа
                                          и его кода}
  End
End.
```

Значения выводятся **парами: символ — код**. В одной строке располагается 10 таких пар. Вся таблица разместится в 24 строках на экране.

Принцип последовательного кодирования алфавитов

В любой кодовой таблице выполняется принцип последовательного кодирования латинского (английского) алфавита и алфавита десятичной системы счисления. Это важное обстоятельство, которое часто учитывается в программах обработки символьной информации.

При выполнении операций отношений, применительно к символьным величинам, учитываются коды этих величин. Чем больше значение кода, тем символ считается больше. Истинными являются следующие отношения: 'A' с 'B', 'Z' > 'Y', 'a' > 'A'. Значение символьной переменной C является прописной (заглавной) латинской буквой, если истинно логическое выражение:

(C >= 'A') and (C <= 'Z')

Значение символьной переменной C является цифрой, если истинно логическое выражение:

(C >= ’0') and (C <= '9')

В латинском алфавите 26 букв. Поэтому разница между кодами букв 'Z' и 'A', а также 'z' и 'a' равна 25.

Пример 2. С помощью датчика случайных чисел заполнить массив **Sim[0. .10]** строчными английскими буквами. Затем массив отсортировать в алфавитном порядке.

```
Uses CRT;
Var Sim: array[0..10] of Char;
    C: Char; i, k: Integer;
Begin
  ClrScr;
  Randomize; {Заполнение массива случайными буквами}
  WriteLn('Исходный массив:');
  For i:= 0 To 10 Do
  Begin
    Sim[i]:=Chr(Random(26)+Ord('a'));
    Write(Sim[i])
  End;
  Writeln;
  {Сортировка методом пузырька}
  For i:=0 To 9 Do
    For k:=0 To 9-i Do
      If Sim[k]>Sim[k+1] Then
      Begin
        C:=Sim[k]; Sim[k]:=Sim[k+1]; Sim[k+1]:=C
      End;
  Writeln('Отсортированный массив:');
  For i:=0 To 10 Do Write(Sim[i]);
End.
```

При тестировании программы было получено:

Исходный массив:

gnkbeqgmsin

Отсортированный массив:

beggikmnnqs

Строки символов

Рассмотрим еще один **структурный тип данных — строковый тип**. Строковый тип данных был введен в Турбо Паскале. Он позволяет программировать обработку слов, предложений, текстов.

Строка — это последовательность символов. Каждый символ занимает 1 байт памяти (код ASCII). **Количество символов в строке называется ее длиной**. Длина строки может находиться в диапазоне от 0 до 255. **Строковые величины могут быть константами и переменными**.

Строковая константа записывается как последовательность символов, заключенная в апострофы. Например:

' Язык программирования ПАСКАЛЬ'

' IBM PC - computer'

'33-45-12'

Строковая переменная описывается в разделе описания переменных следующим образом:

Var <идентификатор>: String[<максимальная длина строки>]

Например:

Var Name: String[20]

Параметр длины может и не указываться в описании. В таком случае подразумевается, что он равен максимальной величине — 255. Например:

Var slovo: String

Строковая переменная занимает в памяти на 1 байт больше, чем указанная в описании длина. Дело в том, что один (нулевой) байт содержит значение текущей длины строки. Если строковой переменной не присвоено никакого значения, то ее текущая длина равна нулю. По мере заполнения строки символами ее текущая длина возрастает, но она не должна превышать максимальной по описанию величины.

Символы внутри строки **индексируются (нумеруются), начиная с единицы**. Каждый отдельный символ идентифицируется именем строки с индексом, заключенным в квадратные скобки. Например:

Name[5], Name[i], slovo[k+1].

Значение индекса может быть задано положительной константой, переменной, выражением целочисленного типа. Оно не должно выходить за границы описания.

Тип **String** и стандартный тип **Char** совместимы: строки и символы могут употребляться в одних и тех же выражениях.

Строковые выражения строятся из строковых констант, переменных, функций и знаков операций. Над строковыми данными допустимы операция сцепления и операции отношения.

Операция сцепления (+) применяется для соединения нескольких строк в одну результирующую строку. Сцеплять можно как строковые константы, так и переменные.

Например:

'ЭВМ'+' 1BM'+' PC'

В результате получится строка:

'ЭВМ IBM PC'

Длина результирующей строки не должна превышать 255.

Операции отношения: =, <, >, <=, >=, < > производят сравнение двух строк, в результате чего получается логическая величина (**true или false**). Операции отношения имеют более низкий приоритет, чем операция сцепления. Сравнение строк производится слева направо до первого несовпадающего символа, и та строка считается больше, в которой первый несовпадающий символ имеет больший номер в таблице символьной кодировки.

Если строки имеют различную длину, но в общей части символы совпадают, считается, что более короткая строка меньше, чем более длинная. Строки равны, если они полностью совпадают по длине и содержат одни и те же символы.

Пример

Выражение:	**Результат:**
'cosm1'<'cosm2'	True
'pascal'>'PASCAL'	True
'Ключ_'<>'Ключ'	True
'MS DOS'='MS DOS'	True

Функции и процедуры

Функция **Copy(S, Poz, N)** выделяет из строки **S** подстроку длиной **N** символов, начиная с позиции **Poz**. **N** и **Poz** — целочисленные выражения.

Пример

Значение *S*:	**Выражение:**	**Результат:**
'ABCDEFG'	Copy(S, 2, 3)	'BCD'
'ABCDEFG'	Copy(S, 4, 4)	'DEFG'

Функция **Concat (SI, S2, . . ., SN)** выполняет сцепление (конкатенацию) строк **S1,..., SN** в одну строку.

Пример

Выражение:	Результат:
`Concat('AA', 'XX', 'Y')`	`'AAXXY'`

Функция **Length (S)** определяет текущую длину строки **S**. **Результат — значение целочисленного типа**.

Пример

Значение *S*:	Выражение:	Результат:
`'test-5'`	`Length(S)`	6
`'(A+B)*C'`	`Length(S)`	7

Функция **Pos (SI, S2)** обнаруживает первое появление в строке **S2** подстроки **S1**. **Результат — целое число, равное номеру позиции**, где находится первый символ подстроки **S1**. Если в **S2** не обнаружена подстрока **S1**, то результат равен 0.

Пример

Значение *S2*:	Выражение:	Результат:
`'abcdef'`	`Pos('cd', S2)`	3
`'abcdcdef'`	`Pos('cd', S2)`	3
`'abcdef'`	`Pos('k', S2)`	0

Процедура **Delete (S, Poz, N)** удаляет **N** символов из строки **S**, начиная с позиции **Poz**.

Пример

Исходное значение:	Оператор:	Конечное значение:
`'abcdefg'`	`Delete(S,3,2)`	`'abefg'`
`'abcdefg'`	`Delete(S,2,6)`	`'a'`

В результате выполнения процедуры уменьшается текущая длина строки в переменной **S**.

Процедура **Insert (SI, S2, Poz)** выполняет вставку строки **S1** в строку **S2**, начиная с позиции **Poz**.

Пример

Начальное *S2*:	Оператор:	Конечное *S2*:
`'ЭВМ PC'`	`Insert('IBM-',S2,5)`	`'ЭВМ IBM-PC'`
`'Рис. 2'`	`Insert('N',S2,6)`	`'Рис. N2'`

Примеры программ обработки строк

Пример 1. Составить программу, формирующую символьную строку, состоящую из **N** звездочек (**N** — целое число, $1 \le N \le 255$).

```
Program Stars;
Var A: String;
    N, I: Byte;
Begin
  Write ('Введите число звездочек');
  ReadLn(N);
  A:='';
  For I:=1 To N Do
    A:=A+'*';
  WriteLn(A)
End.
```

здесь строковой переменной A вначале присваивается значение пустой строки, обозначаемой двумя апострофами (' '). Затем к ней присоединяются звездочки.

Пример 2. В символьной строке подсчитать количество цифр, предшествующих первому символу '!'.

```
Program C;
Var S: String;
    K, I: Byte;
Begin
  WriteLn('Введите строку');
  ReadLn(S);
  K:=0; I:=1;
  While (I<=Length(S)) And (S[I]<>'!') Do
  Begin
    If (S[I]>='0') And (S[i]<='9')
    Then K:=K+1;
    I:=I+1
  End;
  WriteLn ('Количество цифр до символа "!" равно', K)
End.
```

В этой программе переменная **К** играет роль счетчика цифр, а переменная **I** — роль параметра цикла. Цикл закончит выполнение при первом же выходе на символ ' ! ' или если в строке такого символа нет, то при выходе на конец строки. Символ S[I] является цифрой, если истинно отношение: 'O' ≤ S[I] ≤ '9'.

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: §§27,28, задания №5 (§27) и №9 (§28) письменно в тетради.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20___ г.
Номер урока: 32
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Практическая работа №15 «Программирование обработки строк символов».
Тип урока: Урок-практикум.
Цели урока:
Образовательная: формирование знаний учащихся об программировании на языке Паскаль.
Развивающая: развитие алгоритмического мышления, памяти, внимательности.
Воспитательная: воспитывать научное мировоззрение, информационную культуру, расширять кругозор учащихся.
План урока:

1. Организационный момент (1 мин.)
2. Практическая работа (34 мин.)
3. Подведение итогов урока (3 мин.)
4. Домашнее задание (2 мин.)

Приемы, используемые на уроке: фронтальная (беседа), решение проблемных задач, работа в группах (парах).
ТСО и оборудование: компьютеры, проектор, экран, Microsoft PowerPoint.

Ход урока:

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

2. Практическая работа

Ход работы

Задание

Составить на Паскале программу решения поставленной задачи по обработке символьных строк. По возможности, использовать подпрограммы. В последующих задачах подразумевается, что слова в тексте (в строке) отделяются друг от друга пробелами.

1. Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.
2. Дана строка, содержащая английский текст. Найти количество слов, начинающихся с буквы «b».
3. В строке заменить все двоеточия (:) точкой с запятой (;). Подсчитать количество замен.
4. Дана строка. Преобразовать ее, заменив звездочками все двоеточия (:), встречающиеся среди первых n/2 символов, и заменив точками все восклицательные знаки, встречающиеся среди символов, стоящих после n/2 символов. Здесь n - длина строки.
5. В строке удалить символ двоеточие (:) и подсчитать количество удаленных символов.
6. Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобки. Вывести на экран все символы, расположенные внутри этих скобок.
7. Дана строка, содержащая текст. Найти длину самого короткого и самого длинного слов.

3. Подведение итогов урока
Оценки за урок, следующие _____

4. Домашнее задание
Записываем домашнее задание: Повторение §§1-28.
Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20__ г.
Номер урока: 33
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Итоговая контрольная работа по курсу 10 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 10 класса (34 мин.)
3. Подведение итогов урока (3 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 10 класса

3. Подведение итогов урока
Оценки за урок, следующие _____

Комментарии учителя.

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.

Класс: 10
Дата: «___» _____ 20__ г.
Номер урока: 34
Учебник: Семакин И.Г., Залогова Л.А., Русаков С.В. и др.
Тема урока: Повторение материала 10 класса.
Тип урока: Комбинированный урок.
Цели урока:
Образовательные: повторение материала 10 класса.
Воспитательные: способствовать воспитанию информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
План урока:

1. Организационный момент (3 мин.)
2. Повторение материала 10 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 10 класса

3. Подведение итогов урока
Оценки за урок, следующие _____

Урок окончен, до свидания!

Список литературы:

1. Информатика: Базовый уровень: учебник для 10 класса / И.Г. Семакин, Е.К. Хеннер, Т.Ю. Шеина. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2014. – 264 с.: ил.