ИНФОРМАТИКА
И ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Учебник для 9-го класса общеобразовательных средних школ

9

Второе издание

Утверждено Министерством народного образования Республики Узбекистан

Издательско-полиграфический творческий Дом имени Чулпана
Ташкент – 2015

УДК: 372.8:004(075)
ББК 32.81(5У)я721
Б 79

Авторы:
Б.Ж. Болтаев, А.Р. Азаматов, А.Д. Аскаров,
М.К. Садыков, Г.А. Азаматова

Ответственный редактор

Н. Тайлаков *— доктор педагогических наук, профессор.*

Рецензенты:

М. Арипов *— профессор кафедры «Информатика и прикладное программирование», доктор физико-математических наук,*
М. Ташов *— учитель информатики высшей категории общеобразовательной школы № 52 Чустского района Наманганской области.*

Условные знаки:

Запомните

Вопросы и задания

Урок для проведения практической или контрольной работы

«Издано за счет средств Республиканского целевого книжного фонда»

ISBN 978-9943-05-745-6

ГЛАВА I ОСНОВЫ АЛГОРИТМИЗАЦИИ

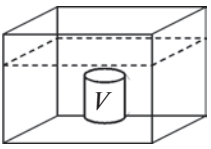
Урок 1. Этапы решения задач на компьютере

В повседневной жизни человеку приходится сталкиваться с решением всевозможных задач. Некоторые из задач связаны с простыми вычислениями, а некоторые со сложными. А при решении некоторых задач придется выполнить некоторую группу операций тысячи раз. И поэтому естественно возникает вопрос: сможет ли помочь в таких случаях компьютер, являющийся покорным и очень быстрым помощником, и если сможет, то как организуется решение задач на компьютере? Перед тем как ответить на этот вопрос, рассмотрим несколько задач и их решения.

Задача 1. Определить значение выталкивающей силы при погружении в воду тела объемом 20 см^3.

Анализируем задачу: из курса физики известно, что погруженное в воду тело вытолкнет объем воды равный своему объему, и на него действует сила равная массе вытолкнутой воды, эта сила называется силой Архимеда.

Чертеж:



Даны:

$V = 20\ см^3 = 20 \cdot \frac{1}{100} \cdot \frac{1}{100} \cdot \frac{1}{100}\ м^3;$

$\rho = 1000\ \frac{кг}{м^3};$

$g = 9{,}81\ \frac{Н}{кг}.$

Определить: $F_A - ?$

Формулы:

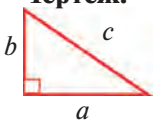
$F_A = \rho \cdot V \cdot g.$

Решение: $F_A = 1000\ \frac{кг}{м^3} \cdot \frac{20}{1000000}\ м^3 \cdot 9{,}81\ \frac{Н}{кг} =$

$= 0{,}1962\ \frac{кг}{м^3} \cdot м^3 \cdot \frac{Н}{кг} = 0{,}1962\ Н.$ **Ответ:** 0,1962 *Н*.

Задача 2. Мухтар начертил красным карандашом на клетчатом листе прямоугольный треугольник с основанием 16 клеток, а его высота равна 3/4 части основания. Определите периметр данного треугольника.

Анализируя задачу, определим, что для решения задачи, во-первых, не имеет значение, каким цветом нарисован треугольник, т.е. эта для нас **«ненужная»** информация, во-вторых, является важной информацией то, что треугольник **прямо**угольный. Если принять во внимание, что длина двух клеток равна 1 см, то в курсе геометрии решение задачи представляется в следующем виде:

Чертеж:	**Даны:**	**Формулы:**
	$a = 16$ клеток $= 8$ *см;* $b = 8$ *см* $\cdot$ 3/4 $= 6$ *см.* **Определить:** $P_{тре} - ?$	Периметр: $P_{тре} = a+b+c$. Теорема Пифагора: $c^2 = a^2+b^2$.

Решение: Из теоремы Пифагора:

$$c = \sqrt{a^2 + b^2} = \sqrt{(8см)^2 + (6см)^2} = \sqrt{100см^2} = 10\ см.$$

Тогда: $P_{тре} = 8\ см + 6\ см + 10\ см = 24\ см$.

Ответ: 24 *см*.

Задача 3. Бехзод прочел четыре страницы и еще четыре строки книги. Каждая строка книги содержит количество символов равное количеству строк на одной странице. Объем прочитанной Бехзодом информации равен 6560 байтам. Определить количество строк, содержащихся на одной странице книги.

Перейдем к анализу задачи.

Исходные данные задачи:

- Бехзод прочел 4 страницы и 4 строки книги;
- объем информации, прочитанной Бехзодом, составил 6560 байтов.
- количество строк на одной странице равно количеству символов одной строки.

Цель задачи:

Определить количество строк, содержащихся на одной странице книги.

Составление уравнения по условиям задачи.

Пусть на одной странице книги количество строк равно x. Следовательно, в каждой строке содержится x символов. Тогда на одной странице книги умещается x^2 символов (т.к. имеется x символов в x строк). По условиям задачи, Бехзод прочел $4x^2 + 4x$ (количество строк на 4-х страницах плюс 4 строки) символов. С другой стороны, количество этих символов равно 6560 (один символ — один байт):

$$4x^2 + 4x = 6560.$$

Уравнение легко приводится в следующее квадратное уравнение: $x^2 + x - 1640 = 0$, т.е. нами было составлено уравнение по условиям задачи.

Последовательность решения данного уравнения.

Последовательность решения квадратного уравнения вам известно:

1) вычислим дискриминант: $D = 1^2 - 4 \cdot 1 \cdot (-1640) = 6561 = 81^2$.

2) так как $D > 0$, получаем следующие решения уравнения:

$$x_1 = \frac{-1-81}{2 \cdot 1} = -41, \quad x_2 = \frac{-1+81}{2 \cdot 1} = 40.$$

Анализ результата:

Уравнение имеет два решения. Однако количество строк в книге не может быть отрицательным, и поэтому, решение уравнения, удовлетворяющее условиям задачи $x = 40$. **Ответ:** 40 строк.

Анализируя решения задач, приведенных выше, видим, что они состоят из следующих этапов:

1. В каждой задаче вначале определяется **постановка задачи**, т.е. исходные данные и цель задачи (конечные величины, которые требуется определить).

2. Для решения задачи составляются **формулы**, иначе говоря, **математические соотношения**.

3. Определяется последовательность выполнения **операций** (формул, соотношений) **для решения задачи** (что отчетливо видно в задачах 2–3).

4. **Получение результата и его анализ.**

Задачи, как приведенные выше, так и другие, можно решить с помощью компьютера. Этап решения включает, кроме вышеприведённых 4 этапов, еще этапы **перевод операций на компьютерный язык** и **ввод программы в память компьютера**:

Первый этап. Постановка задачи	Определение исходных данных и конечных величин задачи.
Второй этап. Составление модели задачи	Изложение задачи на языке математических соотношений, с учетом достижений современной науки.
Третий этап. Составление алгоритма	Составление последовательности указаний для решения задач, исходя из модели задачи.

Четвертый этап. Составление программы	Перевод содержания алгоритма на язык программирования.
Пятый этап. Ввод программы в память компьютера	Ввод программы в память компьютера.
Шестой этап. Получение результата и его анализ	Запуск программы и анализ результатов, исправление ошибок.

Так как каждый из указанных выше этапов рассчитан на определенный объем знаний и опыта, то более подробно они будут рассмотрены в следующих темах.

Вопросы и задания

1. *Какие основные этапы включает в себя решение задач на компьютере?*
2. *С какой целью анализируются результаты?*
3. *Какие ошибки характерны для расчётов, производимых на калькуляторе?*
4. *Определите последовательность выполнения операций для арифметического выражения* $23 + 46 \cdot 3 - 24 : 3$.
5. *Приведите примеры задач на составление уравнения по условиям задачи.*

Упражнения

Проанализируйте условия следующих задач и решите их, разбивая на этапы.

1. Скорость катера в стоячей воде равна 15 км/час. Расстояние, которое катер проплыл по течению реки за 2 часа, равно расстоянию, которое катер проплыл против течения за 3 часа. Найдите скорость течения реки (подсказка: скорость = расстояние/время).

2. Если стороны прямоугольника равны, соответственно, 4 см и 3 см, то определите длину его диагонали (подсказка: диагональ прямоугольника делит прямоугольник на два прямоугольных треугольника, т.е., диагональ окажется гипотенузой).

Урок 2. Модель и ее типы

Интересующие нас и изучаемые нами предметы или процессы называют **объектом**. Примерами объектов служат планеты солнечной системы, спортивные мячи, школьные компьютеры. Однотипные изучаемые объекты имеют общие свойства — **параметр**. Каждый отдельно взятый объект отличается от другого объекта присущими ему свойствами — **значением параметра**. Например, параметрами изучаемых объектов под названием компьютеры могут быть: название фирмы-разработчика, марка основной платы (motherboard), название процессора, скорость процессора (CPU), объем винчестера, объем оперативной памяти (RAM), объем видеопамяти. Тогда значениями параметров отдельно взятого компьютера могут служить: название фирмы-разработчика FUJITSU SIEMENS, марка основной платы D1170, название процессора Pentium IV, скорость процессора 3,06 Ггерц, объем винчестера 160 Гбайт, объем оперативной памяти 1 Гбайт, объем видеопамяти 512 Мбайт.

Если изучаемые объекты – планеты:

Параметры планет	форма	вес	радиус	орбитальная скорость
Значение параметра для **Земли**	шарообразная	$5976 \cdot 10^{21}$ кг	6378 км	30 км/сек

Для объекта под названием мяч:

Параметры мячей	форма	вес	радиус	материал
Значение параметра простого мяча	сферическая	2,2 кг	15 см	резина

Во многих случаях при исследованиях в определенной области изучается не реальный объект, а некоторая его копия. Это связано с тем, что с одной стороны по определенным причинам (неустойчивость молнии, отдаленность Солнца, большие затраты при работе с объектом или угроза для жизни человека и др.) невозможно изучить непосредственно реальный объект, с другой стороны будет достаточно изучить некоторую копию объекта.

Несомненно, в таких случаях копия объекта должна отвечать всем требованиям исследуемой области.

Модель — копия реального **объекта**, отвечающая требованиям исследуемой области.

Слово модель (от латинского **modulus** — мера, норма) хорошо знакома вам по кружкам самолетостроения или кораблестроения. Можно привести много примеров моделей объектов. Например, моделью Земли может стать глобус или топографическая карта; моделью самолета – уменьшенная по размерам копия; моделью автомашины – знакомые вам игрушки; моделью молнии — короткое замыкание в электрической цепи высокого напряжения или сгорание сварочного электрода; моделью человека — его клетки или кукла или фотоснимки; вычислительной моделью человеческого мозга — калькулятор или компьютер.

Если при испытаниях реальный объект и его модель выдают одинаковые результаты, то модель соответствует требованиям исследуемой области.

Например, самолет и являющийся его моделью маленькая копия одинаково реагируют на законы аэродинамики.

В основном, полученные результаты при испытаниях модели верны и для настоящего самолета. Построив проектируемый самолет, ведут испытания в специальных лабораторных устройствах — в аэродинамических трубах, направляющих воздушный поток к самолету. В этом случае лабораторные стенды превращаются в модель атмосферы.

Существуют такие процессы, для которых в качестве модели рассматриваются математические соотношения и формулы. В таких случаях выбранная модель должна охватывать свойства реального объекта, т.е. существенные свойства изучаемого объекта и выбранной модели должны представляться одинаковыми математическими соотношениями и формулами.

Представление параметров изучаемого объекта через математические соотношения, символы и формулы называется **математической моделью**.

Процесс представления изучаемого объекта через математические соотношения, символы и формулы называется математическим моделированием.

На предыдущем уроке задача нахождения числа строк на одной странице книги представлена в виде квадратного уравнения. Следовательно, процесс представления задачи в виде квадратного уравнения — математическое моделирование, а квадратное уравнение — математическая модель задачи.

Также являются математической моделью формулы силы Архимеда, теоремы Пифагора и периметра.

Математическое моделирование с давних пор применяется в астрономии, химии и физике. В качестве примера можно привести открытие планеты Нептун. Французский астроном У. Леверье в 1846 году на основе математических расчетов доказал, что наблюдаемая орбита Урана указывает на наличие по соседству неизвестного небесного тела Солнечной системы.

В том же году итальянский астроном Галилей, опираясь на исследования Леверье, наблюдал неизвестную до этого времени планету Нептун.

Приводим примеры математических моделей химических реакций:

1) реакция соединения хлора с натрием:

$$2Na + Cl_2 = 2NaCl;$$

2) реакция получения серы из природного газа:

$$2H_2S + + O_2 = 2H_2O + 2S.$$

Математической моделью физических явлений могут служить следующие примеры:

1) второй закон Ньютона о силе, воздействующей на тело:
$F = ma$, где m — масса тела, a — ускорение;

2) закон всемирного тяготения Ньютона: $F = G\frac{m_1 m_2}{R^2}$, где m_1, m_2 — массы взаимодействующих тел, R — расстояние между ними, G — гравитационная постоянная.

В настоящее время, применив математическое моделирование в химии, биологии, медицине, экономике и других научных дисциплинах, получают интересные научные результаты.

По выбору способа представления объектов, модели принято делить на три типа, как представлено в нижеследующей схеме:

1. Абстрактная модель в свою очередь делится на две группы: **математические** и **экономико-математические** модели.

Математические модели состоят из математических соотношений, формул и математико-логических параметров взаимосвязи строения и закономерностей объекта. Примеры математической модели приведены в предыдущих уроках.

Экономико-математические модели начали развиваться с XVIII века. В работе Ф. Кене «Экономические таблицы» впервые предпринята попытка с помощью модели представить процесс формирования социального производства.

В настоящее время с помощью экономических моделей исследуются общие законы экономического развития. Сложные экономические модели применяются для прогноза и анализа динамики различных показателей, а именно: национального дохода, потребления, занятости, финансовых накоплений, инвестиционного климата и т.п. Переход независимого Узбекистана на рыночную экономику на основе **5 принципов** составляет основу экономико-математической модели (запомните эти принципы).

2. В физических моделях природа и структура объекта сохраняются, но отличаются от оригинала в количественном соотношении (размерами, скоростью и др.). Примерами служат модели самолета, корабля, автомобиля, поездов и др.

3. Биологическая модель применяется для моделирования биологических структур, функций и процессов разных «живых» объектов или их частей (клетка, организм и т.д.). Биологическая модель предоставляет возможность исследовать определенное поведение или заболевания на лабораторных животных. Например, препарат от вредного вируса тестируется не на человеке, а берется малая порция

крови человека и тестируется в его крови. Ниже рассмотрим примеры математической модели экономических, физических и биологических процессов.

Задача 1. Составить математическую модель, отражающую состояние идеального газа объемом V, температурой T и давлением p.

Решение задачи можно представить по формуле Клайперона, т.е. объем, температура и давление идеального газа связаны следующим образом:

$$\frac{pV}{T} = \text{const}.$$

Данная формула явно показывает, что изменение температуры идеального газа является причиной изменения давления или объема.

Задача 2. Составьте модель процесса роста цветка.

Из курса ботаники вам известно, что для жизнеобеспечения и роста растений необходимы воздух, свет, вода и питательные вещества. Их количественные соотношения отличаются для разных растений. Например, некоторые цветы хорошо растут в темноте и сухой комнате, а другие требуют свет и влажность. И поэтому искомая модель выражается через следующую систему уравнений:

$$\begin{cases} T = T_0 \cdot (1 + \alpha t); \\ I = I_0 \cdot (1 + \beta t); \\ H = H_0 \cdot (1 + \gamma t), \end{cases}$$

где t — время, T — температура воздуха, I — освещенность, H — содержание влаги в растении, α, β, γ — постоянные величины, соответствующие температуре, освещенности, влажности.

Изучая рассмотренные модели, можно сказать, что недостаточно для математического моделирования лишь только математических знаний, необходимы еще глубокие знания в смежных областях науки.

Вопросы и задания

1. *Что понимается под термином «объект»?*
2. *Расскажите о параметре объекта и о значении параметра. Приведите примеры.*
3. *Что понимается под термином «модель»?*
4. *Приведите примеры объектов и их моделей.*
5. *Дайте определение математической модели. В каких областях применяются математические модели?*
6. *Чем отличается математическая модель от других моделей?*
7. *Как была открыта планета Нептун?*
8. *Приведите примеры применения математической модели в химии и физике.*

9. Перечислите типы моделей.
10. Какие абстрактные модели вы знаете?
11. Расскажите об экономико-математической модели.
12. Какие физические модели вы знаете?
13. Расскажите о необходимости создания биологической модели.

Упражнения

Напишите параметры и значения параметров следующих объектов.

1. Объект: области (подсказка: название, площадь, численность, основной экономический продукт, ...).
2. Объект: одноклассники (подсказка: пол, рост, цвет волос, вес, цвет глаз, ...).
3. Объект: книги (подсказка: название, объем, цвет, вес, ...).

Урок 3. Повторение тем «Этапы решения задач на компьютере» и «Типы моделей»

1. Проанализируйте условия следующей задачи и решите, разбивая на этапы.

А. Вычислите гипотенузу прямоугольного треугольника с катетами, равными ***a*** и ***b***.

Б. Вычислите площадь прямоугольного треугольника с катетами, равными ***a*** и ***b***.

В. Вычислите высоту равностороннего треугольника со стороной ***a***.

2. Напишите параметры и значения параметров следующих объектов.

а) объект: колледжи вашей области или города (подсказка: название, год постройки, направления, количество принимаемых учеников, ...).

б) объект: автомобили, производимые заводом Асака (подсказка: марка, год начала производства, количество, цвета, ...).

3. Составьте модели и решите следующие задачи.

А. Составьте модель, представляющую положение вклада, вложенного в банк B сумов с A процентной ставкой годового дохода, через M лет.

Подсказка. За первый год вкладчик получит доход $\frac{B}{100} \cdot A$ сумов. Тогда общая сумма вклада к концу года составит $\frac{B}{100} \cdot A + B = B \cdot (\frac{A}{100} + 1)$ сумов. К концу второго года вкладчик получит доход в размере $B \cdot (\frac{A}{100} + 1) \cdot \frac{A}{100}$ сумов. И поэтому общая сумма

вклада к концу второго года составит $B \cdot (\frac{A}{100}+1) \cdot \frac{A}{100} + B \cdot (\frac{A}{100}+1) =$

$= B \cdot (\frac{A}{100}+1) \cdot (\frac{A}{100}+1) = B \cdot (\frac{A}{100}+1)^2$ сумов.

Вычислите общую сумму вклада к концу третьего и четвертого года и обобщите полученные формулы.

Б. Самолет пролетел расстояние 2100 км между городами A и B за 3 часа, расстояние 4800 км между городами B и M за 6 часов. Определите среднюю скорость самолета (подсказка: средняя скорость =(1-путь + 2-путь)/(1-время + 2-время)).

Урок 4. Понятие алгоритма

В своей жизни человек ставит перед собой цель, суть которой – решить те или иные задачи. Обычно, основываясь на своем жизненном опыте и приобретенных знаниях, он приводит в определенный порядок последовательность необходимых действий для достижения своей цели. Примеров тому много.

Пример 1. Пусть наша цель — заварить чай. Чтобы справиться с поставленной задачей, надо выполнить ряд последовательных действий:

1) открыть крышечку чайника;
2) ополоснуть чайник кипятком;
3) насыпать в чайник чайную ложку заварки;
4) залить чайник кипятком;
5) закрыть чайник крышечкой;
6) накрыв чайник полотенцем, дать чаю завариться в течение пяти минут.

Пример 2. Требуется определить количество плиток размером 12×25 (ширина 12 см и длина 25 см) для того, чтобы покрыть дорожку шириной в N метров и длиной M метров. Человек, знакомый с основами геометрии, решение задачи сводит к следующей последовательности действий:

1) определить площадь $S_{дор}$ дорожки в сантиметрах;
2) определить площадь $S_{плит}$ одной плитки в сантиметрах;
3) количество плиток $S_{кол}$ определить как отношение площади дорожки к площади одной плитки.

Последовательность этих действий можно выразить с помощью следующего математического выражения:

$$S_{кол} = \frac{S_{дор}}{S_{плит}} = \frac{N \cdot 100 \cdot M \cdot 100}{12 \cdot 25}.$$

Пример 3. Выполнить действие: 19632107 + 19702202. Как выполнили бы вы это действие? Да, правильно, вы знаете правила действий над числами «столбиком» и поступите почти следующим образом:

1) запишите второе число под первым, чтобы совпали разряды чисел;

2) сложив цифры первых разрядов чисел, цифру первого разряда результата запишите под цифрой первого разряда, а цифру второго разряда результата сохраните в уме;

3) сложив цифры вторых разрядов чисел и сохраненную в уме цифру, цифру первого разряда результата запишите под цифрой второго разряда, а цифру второго разряда результата сохраните в уме; правило 3 пункта повторяется для цифр следующих разрядов. Эти действия вам хорошо знакомы в следующем виде:

$$\begin{array}{r} 19632107 \\ +\underline{19702202} \\ 39334309 \end{array}$$

Выполнение кем-нибудь последовательности указаний или действий, другими словами — команд, в вышеизложенных задачах приводит к намеченной цели. Из встречающихся каждый день и каждый час разных правил, правила, требующие исполнить последовательность действий, приводящих к некоторому необходимому результату, определяются одним из основных понятий информатики — словом **алгоритм**.

Слово «алгоритм» происходит от латинской формы имени нашего соотечественника, жившего в IX веке (783—850), всемирно известного своими сокровищами научных работ, великого астронома, математика и географа Абу Абдуллах Мухаммед ибн Муса **аль-Хорезми**.

Математический трактат аль-Хорезми, посвященный алгебре, был переведен в XII веке с арабского языка на латынь в Испании. Единственная копия упомянутого перевода, сделанная в XIV веке, хранится в настоящее время в библиотеке Кембриджского университета.

Трактат начинается словами **«Dixit Algorithmi»**, т.е. словами «И сказал аль-Хорезми».

Каждое из указаний или предписаний, другими словами — команд алгоритма подразумевает выполнение некоторого **действия**. Объект, который способен выполнить действия, предписываемые алгоритмом, связан с понятием **исполнитель**.

Каждый алгоритм — это правило, определяющее последовательность действий, а эта «цепочка» действий приведет от исходных данных к искомому результату. Вся цепочка действий называется **алгоритмическим процессом**, а каждое действие — шагом алгоритма.

Под словом **алгоритм** понимается последовательность предписаний для исполнителя, направленная на достижение намеченной цели.

Таким образом, последовательность указаний или команд в вышеизложенных задачах и есть **алгоритм**, а исполняющее их лицо является **исполнителем**. Последовательность указаний в первом примере называют «Алгоритмом заварки чая». Отсюда можно сделать вывод о том, что человек, наметивший для себя ту или иную цель, как исполнитель выполняет серию алгоритмов. Многие алгоритмы станут привычным делом для человека.

Например, приготовление и прием пищи, умывание, порядок одевания, переход с одного места на другое и т.д.

Нетрудно представить какие будут последствия, если нарушить порядок действий в алгоритме. Для примера можно поменять местами в «Алгоритме заварки чая» первое и третье указание и выполнить.

Обычно, для того, чтобы указания были понятными исполнителю, они составляются из простых действий.

Например, первое указание алгоритма во втором примере можно разбить на три простых указания:

1 а) перевести ширину дорожки N в сантиметры;

1 б) перевести длину дорожки M в сантиметры;

1 в) подсчитать площадь дорожки $S_{дор}$ в сантиметрах.

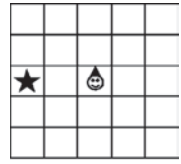
Естественен вопрос: является ли только человек исполнителем алгоритма? На этот вопрос ответим так:

Исполнитель алгоритма — это некоторая абстрактная или реальная (техническая или биологическая) система, способная выполнить действия, предписанные алгоритмом.

Множество указаний или команд, которые может выполнить исполнитель, называется **системой команд исполнителя** (сокращенно **СКИ**). Например, команда «Найти квадратный корень из 16» не входит в систему команд для ученика 2 класса, но входит в систему команд для ученика

8 класса. Следует подчеркнуть, что в информатике универсальным исполнителем алгоритмов является **компьютер.**

Чтобы понять систему команд исполнителя, рассмотрим следующую задачу.



Пример 4. Для Колобка «передняя» клетка – это клетка, на которую указывает его шапочка, например, когда он поворачивается вправо, окажется в положении ◑. Колобок может переходить на одну клетку вперед или поворачиваться внутри клетки влево или вправо. Колобок может проходить через одну клетку несколько раз. Если Колобок сможет перейти из клетки, в которой он находится, в клетку, отмеченную знаком ★, то напишите последовательность нужных указаний. По условию задачи можем написать систему команд Колобка (СКК), т.е. **СКК={вперед; влево; вправо}**. Теперь как решение задачи можно выбрать один из следующих алгоритмов:

Число шагов	**1-алгоритм**	**2-алгоритм**	**3-алгоритм**
1	1) влево;	1) вправо;	1) вперёд;
2	2) вперёд;	2) вправо;	2) влево;
3	3) вперёд.	3) вправо;	3) вперёд;
4		4) вперёд;	4) вперёд;
5		5) вперёд.	5) влево;
6			6) вперёд.

Следовательно, алгоритм, дающий **решение задачи**, **может** быть **неединственным**. Из вышесказанных и рассмотренных примеров следует, что обычно исполнитель может не знать о цели алгоритма. Например, заранее не известно о цели следующего алгоритма:

1) определить натуральные числа N и M;

2) приравнять S к нулю;

3) значение максимального из чисел N и M принять равным разнице максимального и минимального из чисел N и M и к S добавить 1;

4) если числа N и M положительные, перейти к пункту 3, в противном случае перейти к пункту 5;

5) в качестве ответа принять значение S.

Между тем, приведенный выше алгоритм решает следующую конкретную задачу.

Пример 5. Прямоугольник задан сторонами, равными натуральным числам N и M. Сколько квадратов можно отрезать из прямоугольника, если каждый раз отрезается самый большой по размеру квадрат? Знакомясь с основными понятиями, как **алгоритм, исполнитель алгоритма, система команд исполнителя,** связанные с этапом решения задач с помощью компьютера, приходим к заключению: **исполнитель управляется через алгоритм**.

Вопросы и задания

1. *Что такое алгоритм?*
2. *Расскажите о происхождении термина алгоритм.*
3. *Найдите примеры алгоритма из школьной жизни.*
4. *Составьте алгоритм нахождения нужной темы из учебника.*
5. *Составьте алгоритм приготовления плова.*
6. *Составьте алгоритм запуска компьютера.*
7. *Что понимается под термином «исполнитель алгоритма»?*
8. *Какие команды исполнитель не сможет выполнить?*
9. *Приведите примеры системы команд исполнителя.*
10. *Составьте невыполнимый алгоритм для вашего одноклассника.*
11. *Можно ли назвать алгоритмом последовательность следующих указаний, и если да, то какую цель они преследуют?*

    1) набрать из реки ведро воды;
    2) вылить из ведра воду в реку;
    3) повторить в пункт 1.

Урок 5. Основные свойства алгоритма

На предыдущем уроке вы ознакомились с понятиями «алгоритм» и «исполнитель алгоритма». Теперь расскажем об основных свойствах алгоритма.

1. Понятность. Чтобы алгоритм был понятен исполнителю, необходимо знать возможности исполнителя. Если исполнителем алгоритма является человек, то алгоритм нужно составить, исходя из его возможностей. В этом случае, исходя из намеченной цели и алгоритма, необходимо учесть язык понимания человека, жизненный опыт, профессиональные навыки, возраст, не говоря уже о физическом состоянии и т.д. Если в роли исполнителя выступает техническое средство (например, компьютер, электронные часы, станки), тогда алгоритм нужно составить, исходя из возможностей предоставленного оборудования.

Таким образом, задаваемое каждое указание должно принадлежать системе команд исполнителя, т.е. исполнитель алгоритма должен знать, как выполнить эти указания.

2. Определенность. Все указания или команды алгоритма должны быть однозначными и точными. Например, указания «положить немного соли» (сколько: чайную ложку или один стакан?), «налить необходимое количество воды» (необходимое, это сколько: 1 литр или 100 литров или может быть тонна), «написать сочинение» (по какой теме?) приводят к различным (иногда к нежелательным) результатам.

Отсюда следует, что благодаря свойству определенности выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

3. Дискретность (прерывность, раздельность). Алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов). Это свойство ярко выражено во всех ранее приведенных алгоритмах.

4. Результативность (конечность). В описании алгоритма использована фраза «для достижения намеченной цели». Намеченные цели можно увидеть во всех задачах, например, заварить чай, посчитать число плиток, вычислить сумму. Это связано со свойством **результативности** алгоритма. Суть этого свойства состоит в том, что исполнение любого алгоритма за конечное число шагов должно привести к некоторому решению. Следует отметить, что алгоритм может и не привести к намеченной цели. Причиной тому могут быть неправильное составление алгоритма или другие ошибки. С другой стороны, поставленная задача может не иметь положительного решения. Однако отрицательный **результат** так же принимается как результат.

Пример 1. Решить квадратное уравнение $x^2 + x + 1 = 0$.

Пользуясь алгоритмом решения квадратного уравнения вида $ax^2 + bx + c = 0$ ($a \neq 0$), приведенного ниже, определим, что заданное уравнение не имеет решения. Как вы знаете, это тоже результат.

1) определить значения a, b, c;

2) вычислить дискриминант: $D = b^2 - 4ac$;

3) если $D < 0$, то принять, что уравнение не имеет решения и перейти к пункту 6;

4) если $D = 0$, то принять, что единственное решение равно $-\frac{b}{2a}$ и перейти к пункту 6;

5) принять, что первое решение равно $\frac{-b-\sqrt{D}}{2a}$, второе решение равно $\frac{-b+\sqrt{D}}{2a}$;

6) завершить.

Если вы заметили, отрицательность и равенство нулю дискриминанта проверено, но не проверена положительность. Подумайте о причине!

Выходит, что алгоритм состоит из конечного числа шагов и должен всегда приводить к некоторому результату.

5. Массовость. Это означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторо-

го класса задач, различающихся лишь исходными данными. Рассмотренный алгоритм решения квадратного уравнения вида приведет к результату для любых чисел *a, b* и *c*. Значит, справедливо свойство массовости алгоритма. Приведенный ниже **алгоритм Евклида** для нахождения наибольшего общего делителя (НОД) двух натуральных чисел также справедлив для всех натуральных чисел.

Пример 2. Найти НОД натуральных чисел N и M.

1) определить натуральные числа N и M;

2) если $N = M$, то принять N как результат и перейти к пункту 4;

3) значение максимального из чисел N и M принять равным разнице максимального и минимального из этих чисел и перейти к пункту 2;

4) завершить.

В заключении отметим, что при выполнении вышеупомянутых свойств последовательность указаний становится алгоритмом и приводит к некоторому (положительному или отрицательному) результату.

Вопросы и задания

1. *Какими свойствами обладают алгоритмы?*
2. *Приведите примеры последовательности указаний, для которых выполняется и нарушается свойство понятность.*
3. *К какой системе должны относиться указания, чтобы они были понятны исполнителю?*
4. *Какое из свойств существенно для механического исполнения алгоритма исполнителем?*
5. *Разъясните на примерах свойство дискретности алгоритма.*
6. *Разъясните на примерах свойство результативности алгоритма.*
7. *Приведите примеры последовательности указаний, для которых нарушается свойство результативности.*
8. *Разъясните на примерах свойство массовости алгоритма.*
9. *Используя алгоритм Евклида, получите несколько результатов.*

Урок 6. Повторение тем «Понятие алгоритма» и «Основные свойства алгоритма»

1. Какие из следующих указаний вы не сможете выполнить как исполнитель и почему?

А. Поднять камень весом 200 кг. Б. Умножить 7 на 2.

В. Считать с 1 по 31622400000.

2. Определите простые команды исполнителю алгоритма для достижения намеченной цели, т.е. систему команд исполнителя.

А. Если открытая дверь находится слева в 5 шагах от исполнителя, то цель «выход через дверь».

Б. Если исполнитель находится возле крана и цилиндрического стакана, то цель «набрать полстакана воды».

В. Вычислить числовое выражение: 44·15+12·15:20−43.

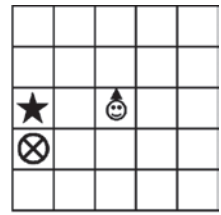
3. С помощью заданных команд составьте алгоритм, приводящий к решению задачи.

А. Старинная задача под названием «Волк, коза и капуста». Крестьянин стоит на левом берегу реки с волком, козой и капустой. Ему надо перевезти все это на правый берег. Но его лодка слишком мала: он может взять только одного пассажира — либо волка, либо козу, либо капусту.

И еще — если на одном берегу оставить волка и козу, то волк съест козу, а если оставить козу и капусту, то коза съест капусту. Только в присутствии крестьянина они не трогают друг друга. Система команд крестьянина такова:

{перевези козу; перевези волка; вернись с козой; перевези капусту; перевези козу}.

Б. Для Колобка «передняя» клетка — это клетка, на которую указывает его шапочка.



Например, когда он поворачивается вправо, окажется в положении ⊙►. Колобок может переходить на одну клетку вперед или поворачиваться внутри клетки вправо, т.е. может выполнить команды **{вперед; вправо}**.

Колобок может проходить через одну клетку несколько раз, но не может пройти клетку с преградой вида ⊗.

Если Колобок сможет перейти из клетки, в котором он находится, в клетку, отмеченной знаком ★, то напишите последовательность нужных указаний.

Урок 7. Способы представления алгоритмов

На предыдущих уроках алгоритмы были представлены в словесном виде. Следует отметить, что существуют различные способы представления алгоритмов. На практике наиболее распространены следующие способы представления алгоритмов.

1. Словесный способ представления алгоритма.

Рассмотренные ранее примеры алгоритмов излагались словами на естественном человеческом языке (например, алгоритм заварки чая или алгоритм сложения). В этом способе представления алгоритмов указания исполнителю задаются словесными фразами.

К примеру для исполнителя с емкостями объемом **A** литров и **B** литров, находящийся возле большого водоема, можно рассмотреть систему команд **{наполни A; наполни B; перелей из A в B; перелей из B в A; вылей из A; вылей из B}**. Целью задач, присущее этому исполнителю, является получение с одной из емкостей A или B необходимое количество воды, которое нужно отмерить.

Задача 1. Составить алгоритм для исполнителя с **A=3** и **B=5**, чтобы отмерить **1 литр** воды.

Алгоритм, приводящий к решению этой задачи, удобно составить словесным способом:

Шаг	**Команда**	**в емкости A**	**в емкости B**
1	наполни A;	3 литра	0 литров
2	перелей из A в B;	0 литров	3 литра
3	наполни A;	3 литра	3 литра
4	перелей из A в B.	1 литр	5 литров

2. Представления алгоритма через формулы.

Этот способ представления алгоритмов часто применяется в математике, физике, химии, биологии. Напомним, что словесный алгоритм решения примера 2 урока 4 был представлен через формулу. На этом примере алгоритмом служит порядок выполнения арифметических действий «+», «−», «×», «:». Следующее решение квадратного уравнения вида $ax^2 + bx + c = 0$ $(a \neq 0)$, рассмотренного на 5 уроке, также служит примером способа представления алгоритмов через формулы:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

3. Табличный способ представления алгоритма.

Такой способ представления алгоритмов вам хорошо известен. Например, расписание уроков в школе, таблица умножения Пифагора, таблица выигрышных лотерейных билетов, таблица химических элементов. Использование этих таблиц требует применения определенного алгоритма.

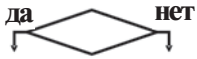
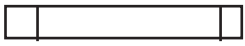
Для рисования графика функции составляется таблица зависимости значения функции от аргумента. Это также служит примером табличного способа представления алгоритма. Например, по урокам математики вы знакомы со следующей таблицей, в которой приведены некоторые точки прохождения исполнителя, движущегося по алгоритму $\mathbf{y = x^2}$:

x	−3	−2	−1	0	1	2	3
y	9	4	1	0	1	4	9

4. Графический способ представления алгоритма.

Этот способ представления алгоритмов известен вам через графики функций в математике, схемы расположения домов в городском квартале или схемы движений автобусов.

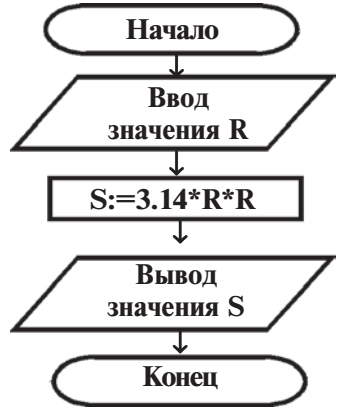
Еще один удобный графический способ представления алгоритма для изучения основ алгоритмизации это **блок-схемы**. Блок-схемы составляются из блоков, т.е. из специальных геометрических фигур, каждая из которых соответствует выполнению одного или нескольких указаний алгоритма. **Блоки** соединяются линиями переходов, определяющими очередность выполнения действий.

	Указывает на начало и конец алгоритма
	Ввод-вывод данных в общем виде
	Вычислительное действие или последовательность действий
	Проверка условий
	Начало цикла
	Вычисления по подпрограмме
	Линия перехода в блок-схеме
:=	Присвоение значения

Задача 2. Составить алгоритм нахождения площади круга с радиусом *R*.

Составим алгоритм двумя способами: словесным и графическим.

1) начало;
2) определить значение радиуса R;
3) умножить R на R и 3,14 и принять за S;
4) вывести S как ответ;
5) завершить.



5. Программный способ представления алгоритма.

Известно, что компьютер работает и управляется посредством программ. Вы уже работали с прикладными **программами** MS Word, MS Paint и MS Excel. Следует отметить, что каждая из этих прикладных **программ** является очень длинным и сложным алгоритмом. Следовательно, **их нужно составить на языке, понятном для исполнителя алгоритма**, т.е. **для компьютера**.

Алгоритм, составленный на языке, понятном для компьютера называется **программой**. Понятный для компьютера язык называется **языком программирования**. В настоящее время в мире насчитываются тысячи языков программирования и их непрерывно развивают. В настоящее время широко распространены и удобны для изучения языки программирования **BASIC, Pascal, VBA, Delphi, C, C++** .

Вопросы и задания

1. *Расскажите о способах представления алгоритма.*
2. *Приведите примеры словесного способа представления алгоритма.*
3. *В каких предметах предпочтительнее представить алгоритм формулами?*
4. *Приведите примеры из физики, в которых алгоритм представляется через формулы.*
5. *Приведите примеры табличного способа представления алгоритма.*
6. *Приведите примеры графического способа представления алгоритма.*
7. *Что такое блок-схема?*

Упражнения

1. Составьте словесный алгоритм для записи столбиком слов текста **«Узбекистан — государство с великим будущим!»** графической программой MS Paint.

2. Составьте удобным способом алгоритм для записи фразы **«Узбекистан — отчизна моя!»** объектом WordArt программы MS Word.

3. Составить алгоритм нахождения НОК (наименьшее общее кратное) двух заданных натуральных чисел.

Урок 8. Практическое занятие по теме «Способы представления алгоритмов»

1. Составьте словесный алгоритм для следующих задач.

А. Составить алгоритм вычисления значения функции $y = 23 \cdot x - 1963$ для заданного значения x.

Б. Система команд исполнителя состоит только из указаний **{прибавь 5; вычти 3}.** Составить алгоритм, с помощью которого исполнитель получит из числа 0 число 11.

В. Система команд исполнителя состоит только из указаний **{прибавь 1; умножь на 2}**. Составить три разных алгоритма, с помощью которых исполнитель получит из числа 0 число 17.

Г. Составить алгоритм для водолея при A=5 и B=8, чтобы отмерить 4 литра воды.

Подсказка. Составление таблицы нижеследующего вида при решении задач *Б* и *В* дает возможность увидеть результат выполнения каждого шага алгоритма:

Шаг	Команда	Результат
0	–	0
1		
2		

2. Составьте алгоритмы для следующих задач с помощью блок-схем.

А. Составьте алгоритм для нахождения стороны квадрата, вписанного в окружность с радиусом *R*.

Б. Даны три монеты. Одна из них поддельная и тяжелая. Для взвешивания используются весы без гири с двумя чашками. Составьте алгоритм для нахождения поддельной монеты.

В. Даны три монеты. Одна из них поддельная и отличается весом (неизвестно, что она тяжелее или легче других). Для взвешивания используются весы без гири с двумя чашками. Составьте алгоритм для нахождения поддельной монеты наименьшим взвешиванием.

Урок 9. Основные типы алгоритмов

Каждый алгоритм по своей логической структуре, т.е. по своему исполнению делится на три типа: **линейные (последовательные), разветвляющиеся и циклические**.

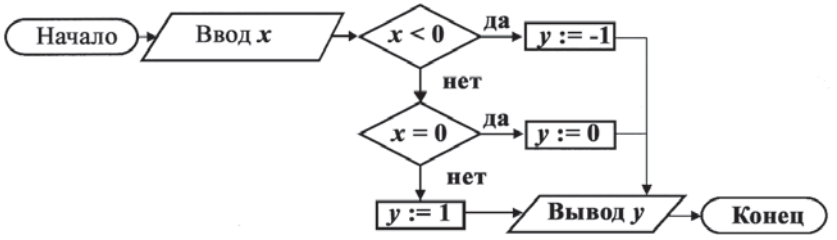
Линейные алгоритмы. Алгоритмы, все команды которых выполняются последовательно по мере расположения, называются **линейными**. «Алгоритм заварки чая», вычисления площади круга служат примерами таких алгоритмов.

Однако в повседневной жизни многие процессы управляются под влиянием определенных условий.

Разветвляющиеся алгоритмы. Алгоритмы, в которых совершается одна или другая последовательность действий в зависимости от условий, называются **разветвляющимися**.

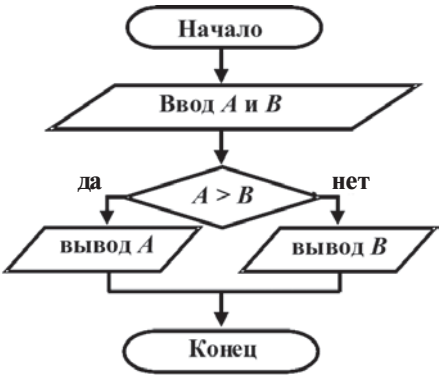
Рассматриваемый тип алгоритма встречается в повседневной жизни на каждом шагу. Выход зависит от того, открыта дверь или нет; прием пищи человеком зависит от того, голоден он или нет или от типа пищи; выбор одежды зависит от погоды; выбор для передвижения транспортного средства зависит от финансовых возможностей и т.д. Следовательно, разветвляющиеся алгоритмы отличаются от линейных алгоритмов возможностью выбора. Примерами разветвляющихся алгоритмов служат алгоритм решения квадратного уравнения, алгоритм нахождения НОД двух натуральных чисел.

Пример 1. Представим разветвляющийся алгоритм вычисления функции $y = \begin{cases} -1, \text{ если } x < 0 \\ 0, \text{ если } x = 0 \\ 1, \text{ если } x > 0 \end{cases}$ в виде блок-схемы:



Пример 2. Составить алгоритм нахождения большего из двух заданных чисел A и B.

1) начало;
2) ввод A и B;
3) если A > B, то
   перейти в пункт 4,
   в противном случае
   перейти в пункт 5;
4) принять A как ответ
   и перейти в пункт 6;
5) принять B как ответ;
6) конец.



Из этого примера можно сделать вывод: если выполняется условие $A > B$, то не рассматривается указание пункта 5, в противном случае, т.е. если $A \le B$, то не рассматривается указание пункта 4.

Этот алгоритм дает возможность наглядно представить разветвляющиеся алгоритмы.

Повторяющиеся (циклические) алгоритмы. При анализе задач можно увидеть, что некоторые команды алгоритма повторно выполняются исполнителем. Например, в задаче вырезания наибольшего по размеру квадрата (урок 4, пример 5), в алгоритме Евклида (урок 5, пример 2). В повседневной жизни человека также встречаются повторяющиеся процессы.

Например, уроки повторяются каждую неделю, каждое утро приходится завтракать или ехать в школу и т.д.

Алгоритмы, в которых некоторые указания выполняются повторно, называются **циклическими**.

Циклические алгоритмы отличаются от других наличием команд вида «I := I + 1», «S:= S + I» или «P := P*I» (* — знак умножения). Чтобы понять смысл таких команд рассмотрим несколько шагов цикла. По установленной традиции считается, что при суммировании исходное значение S:= 0 (первая буква от английского слова SUM, т.е. сумма), а при умножении − P: = 1 (первая буква от английского слова, PRODUCT, т.е. произведение). Потому что эти значения, т.е. 0 и 1, соответственно, не влияют на значения результата суммирования и умножения:

1-шаг. Пусть I := 1, тогда S := S + I = 0 + 1 = 1, P := P*I = =1*1 = 1;

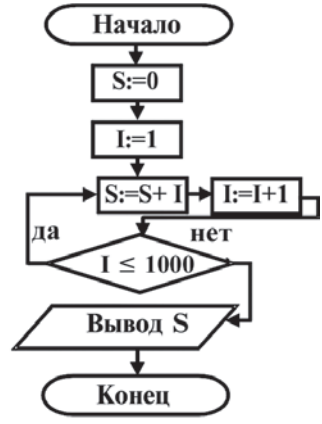
2-шаг: I := I + 1 = 1 + 1 = 2, S := S + I = 1 + 2 = 3, P := P*I = =1*2 = 2;

3-шаг: I := I + 1 = 2 + 1 = 3, S := S + I = 3 + 3 = 6, P := P*I = =2*3 = 6;

4-шаг: I := I + 1 = 3 + 1 = 4, S := S + I = 6 + 4 = 10, P := P*I = =6*4 = 24.

Пример 3. Составить алгоритм суммирования чисел от 1 по 1000, т.е. S = 1+2+3+...+1000.

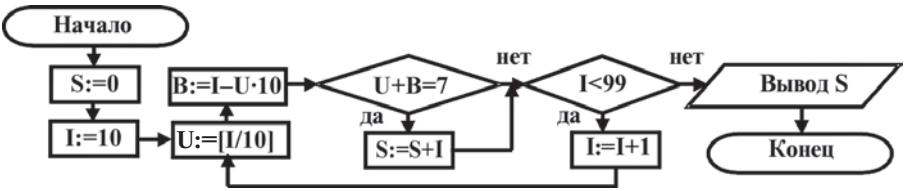
1) начало;
2) принять S равным 0 (т.е. S:=0);
3) принять I равным 1 (т.е. I:=1);
4) принять S равным сумме S и I (т.е. S:=S+I);
5) принять I равным сумме I и 1 (т.е. I:=I+1);
6) если I ≤ 1000, то перейти в пункт 4;
7) принять S как ответ;
8) завершить.



Чтобы показать соответствие словесного алгоритма с блок-схемой, после словесных указаний в скобке даны комментарии. Обычно, в циклических алгоритмах выражение вида «I := I+1» называется **счетчиком**. Решение этого примера можно представить в виде линейного алгоритма. Для этого достаточно применить тождество 1+2+3+...+N≡N·(N+1):2, верное для любого натурального числа *N* (составьте алгоритм самостоятельно).

В следующих примерах нет такой возможности.

Пример 4. Составьте алгоритм суммирования двузначных чисел, сумма цифр которых равна 7 ([*a*] – целая часть числа *a*, / — знак деления).



Пример 5. Составьте алгоритм, который заставит написать 20 раз фразу **«Родина как святая святых»**. Алгоритм этого примера представим словесным способом.

1) значение I принять равным 1;
2) написать «Родина как святая святых»;
3) принять I равным сумме I и 1;
4) если I ≤ 20, то перейти в пункт 2;
5) завершить.

Если проанализировать рассмотренные алгоритмы, можно увидеть, что они состоят из сплетенных линейных, разветвленных и циклических частей.

Значит, подавляющее большинство алгоритмов в жизни человека представляются в единстве этих трех типов.

Вопросы и задания

1. *Какие алгоритмы называются линейными? Приведите примеры.*
2. *Какие алгоритмы называются разветвляющимися? Приведите примеры.*
3. *Какие алгоритмы называются циклическими? Приведите примеры.*
4. *Объясните отличия линейных, разветвляющихся и циклических алгоритмов.*

5. *Составьте алгоритм нахождения наибольшего среди трех заданных чисел.*

Упражнения

1. Определите тип и результат алгоритмов, приведенных ниже:

а) a:=3; x:=2*a+a*a. a=?, x=?

б) x:=1; x:=x+11, x:=x∗x−4. x=?

в) a:=15; b:=a; a:=a−b. a=?, b=?

г) 1) a:=3;
2) если a>2, то x:=2*a+a*a и перейти в пункт 4, в противном случае перейти в пункт 3;
3) x:=9−a*x;
4) напечатать результат x;
5) завершить.

д) 1) x:=1;
2) если x > 2, то x:=x+11 и перейти в пункт 4, в противном случае перейти в пункт 3;
3) x:=x*x−4;
4) напечатать результат x;
5) завершить.

е) 1) a:=15;
2) b:= a;
3) если a > b, то a:=a−b и перейти в пункт 5, в противном случае перейти в пункт 4;
4) a:=a+b;
5) напечатать результат a, b;
6) завершить.

2. Составьте алгоритм определения знака заданного числа в виде блок-схем.

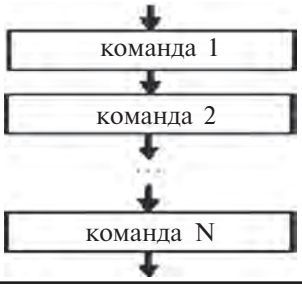
3. Составьте алгоритм вычисления значений функции $y = x^2 - 1$ в виде блок-схем для целых чисел x из [1; 10].

Урок 10. Практическое задание по базовым структурам алгоритма

Как было отмечено, каждый алгоритм можно рассматривать как сплетение линейных, разветвленных и циклических частей.

Поэтому будет целесообразно усвоить нижеследующие базовые структуры. Характерной особенностью базовых структур является наличие в них одного входа и одного выхода.

1. Линейная структура. Эта структура образуется из последовательности таких команд, как ввод значения или соответствующие указания, следующих одно за другим.

Словесный способ	В виде блок-схем
команда 1 команда 2 ... команда N	

Задача 1. Даны числа *a*, *b*, *c*. Составьте алгоритм вычисления половины суммы чисел *a* и *b*, модуль разности чисел *a* и *c*, квадрат произведения чисел *b* и *c*.

Задача 2. Составьте алгоритм вычисления длины окружности, площадь круга и объем шара с радиусом *R* (подсказка: $L=2\pi R$; $S=\pi R^2$; $V=\frac{4}{3}\pi R^3$).

2. Структура **ветвления**. Эта структура обеспечивает, в зависимости от результата проверки условия (да или нет), выбор одного из альтернативных путей работы алгоритма. Структура ветвления существует в двух основных видах.

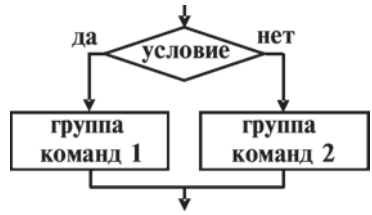
а) если — то:

Словесный способ	В виде блок-схем
если условие **то** группа команд **конец**	

Задача 3. Составьте алгоритм вычисления квадрата и квадратного корня заданного числа *a*, если оно положительное.

Задача 4. Составьте алгоритм, который вычисляет квадратный корень неотрицательных чисел среди заданных чисел *a*, *b*, *c*.

б) если – то – иначе:

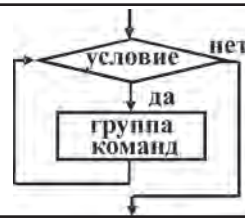
Словесный способ	В виде блок-схем
если условие **то** группа команд 1 **иначе** группа команд 2 **конец**	

Задача 5. Даны числа a, b, c. Составьте алгоритм, который выводит ответ «Да», если выполнено условие $a < b—c$, в противном случае – «Нет».

Задача 6. Даны числа a и b. Составьте алгоритм, который выводит квадрат каждого из них, если их произведение положительное, в противном случае каждое из них, увеличенное на 100.

3. Структура «цикл». Эта структура обеспечивает многократное выполнение некоторой совокупности действий. Структура «цикл» существует в двух основных видах.

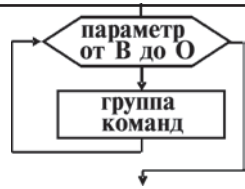
а) пока:

Словесный способ	В виде блок-схем
пока условие группа команд **конец**	

Задача 7. Составьте алгоритм, который вычисляет значения функции $y = ax^2+20$ для всех натуральных значений x меньших заданного числа a.

Задача 8. Составьте алгоритм, который заменит наибольшее из положительных чисел A и B разностью наибольшего с наименьшим среди чисел A и B, пока они не сравняются.

б) параметр от … до … :

Словесный способ	В виде блок-схем
параметр от В до О группа команд **конец**	

Здесь параметр рассматривается как счетчик, и поэтому вместо него можно взять любую букву отличную от *B* и *O*.

Задача 9. Составьте алгоритм для написания фразы **«Узбекистан — государство с великим будущим!»** столько, сколько лет исполнилось независимости Узбекистана в этом учебном году.

Теперь с помощью этих структур можно легко выразить алгоритмы задач, рассмотренных в предыдущих уроках.

Вопросы и задания

1. *Приведите пример структуры, соответствующей линейному алгоритму.*
2. *Какие виды структуры ветвления вы знаете?*
3. *Приведите пример применения удобного вида структуры ветвления.*
4. *Расскажите о структуре «цикл».*
5. *В каких задачах целесообразно применять структуру «пока»?*

Упражнения

1. Составьте алгоритм, который вычисляет расстояние, которое проедет машина со скоростью v км/час за T часов.

2. Пусть R_1, R_2, R_3 — радиусы круга. Составьте алгоритм, который вычислит квадрат совокупной площади круга.

3. Составьте алгоритм, который вычислит модуль разности площадей квадратов со сторонами *a* и *b*.

4. Даны числа *a* и *b*. Составьте алгоритм, который заменит *b* нулем, если число *b* меньше *a*, в противном случае оставит *b* без изменения.

Урок 11. Задания для повторения

1. Расскажите о первых трех этапах на основе решения задачи нахождения длины окружности, вписанной в квадрат со стороной равной *a*.

2. Расскажите о первых трех этапах на основе решения следующей задачи: в 50-литровой емкости содержится 20 литров смеси с 5 килограммами соли. Найдите процентное соотношение соли в смеси, если в емкость добавить еще 10 литров жидкости.

3. Даны числа *a*, *b* и *c*. Составьте алгоритм, который вычислит $y = a^2 - b^2$, если $a + b + c<0$ в противном случае $y = a^2+c^2$.

4. Составьте алгоритм, который вычислит произведение нечетных чисел от − 100 до 50.

ГЛАВА II ОСНОВЫ ПРОГРАММИРОВАНИЯ

Урок 12. Программа и языки программирования

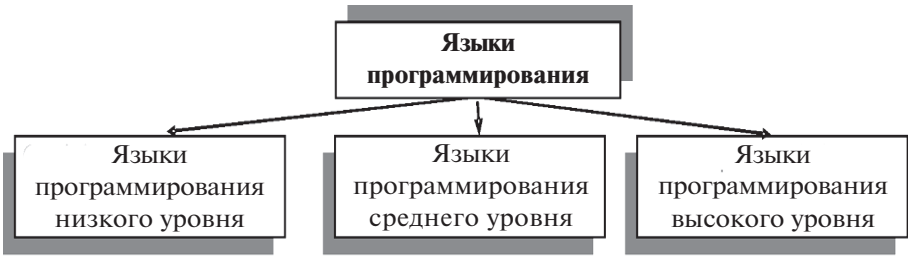
Известно, что эффективное использование компьютера требует неразрывной связи двух частей: технического и программного обеспечения. Стремительное развитие технического обеспечения компьютера влечет за собой непрерывное усовершенствование соответствующего программного обеспечения, и наоборот. Причина этого и так ясна – без соответствующего программного обеспечения компьютер превращается всего лишь в «дорогую игрушку».

На предыдущих занятиях вы ознакомились с такими понятиями, как объект, модель и алгоритм, без которых невозможно решение задач на компьютере. Известно, что для решения задачи на компьютере сначала строится ее модель и алгоритм, затем на основе определенных правил этот алгоритм записывается в виде последовательности команд и указаний, понятных компьютеру.

Таким образом, составленный на понятном компьютеру языке алгоритм называется **программой**, а текст алгоритма — **текстом программы.**

Процесс составления программы называется **программированием**, а человек, составляющий программу — **программистом**. Язык, воспринимаемый компьютером, называется **языком программирования**.

Языки программирования условно можно разделить на три группы:



Языки программирования низкого уровня непосредственно связаны с устройствами компьютера, и команды представляют собой набор чисел (код). Таким образом, составленные программы имеют большой объем и редактировать их слишком трудно. В первых электронных вычислительных машинах («ENIAC», «МЭСМ» и др.) программы для решения задач составлены с помощью таких команд.

Из истории языков программирования. Языки программирования, в основном, развивались после второй мировой войны. Но их история связана с далеким прошлым.

На керамической доске, найденной при археологических раскопках Вавилона и датируемой 1800 годами до нашей эры, приводится алгоритм, связанный с процентом. В ней решена такая задача: если урожай зерна каждый год растет на 20%, то после скольких лет и месяцев урожай возрастет в два раза.

Чарльз Бэббидж

В 1804 году французский изобретатель **Жозеф Мари Жаккар** в производстве тонкой ткани использовал для станков ленту, напоминающую перфокарту.

В 1836 году английский ученый **Чарльз Бэббидж** начал разработку *и теоретически обосновал возможность создания аналитической машины,* являющейся предком современных компьютеров. Отличительные свойства этой машины: работа с помощью программы и запоминание результатов вычисления.

Ада Лавлейс Байрон

В 1843 году английский математик **Огаста Ада Байрон** (Лавлейс) — дочь поэта лорда Байрона подчеркнула, что аналитическая машина должна работать на основе команд. Она написала команды, которые обеспечивают выполнение последовательности команд, пока не выполнены условия. Именно этим она заложила основу языка программирования. Эти и другие открытия послужили основой для разработки языков программирования после создания компьютеров.

С целью облегчить процесс программирования была поставлена задача применения системы команд, близких к человеческому языку. В результате появились **языки программирования среднего уровня** (иногда их называют **ассемблерами**).

К таким языкам относятся **AVTOKOD-BEMSH, AVTOKOD-MADLEN** и другие. Они применялись в электронных вычислительных машинах **BESM-6, Minsk-22, Minsk-32, IBM-360**. Например, команда **ST 5, BSUM** дает указания: число 5 вставить в ячейку с именем BSUM (**ST**-store — вставить).

Команды **языка программирования высокого уровня** состоят из слов и словосочетаний очень близких к человеческому языку. Составить программы на таких языках относительно легче, в этом случае программист не обязан знать архитектуру компьютера и его устройств.

Но программа, написанная на таком языке, не понятна компьютеру, и поэтому применяются специальные программы — **трансляторы** (переводящие программу с языка высокого уровня в коды, воспринимаемые компьютером).

В последние годы были разработаны очень много языков программирования высокого уровня, к которым относятся **Pascal, Ada, KARAT, C++, Delphi, Visual Basis Application, Java**. Разрабатываемые в настоящее время языки программирования характеризуются решением задач некоторого направления, и их называют **объектно-ориентированными языками программирования**.

В следующей таблице приведена краткая информации из истории развития языков программирования.

Язык программирования	**Год разработки**	**Язык программирования**	**Год разработки**
Plankalkyul	1946	Logo	1967
Короткий код	1949	Алгол 68	1968
Assembler «Edsak», AO	1950	APL	1969
Автокод «Madlen»	1953	Paskal	1970
Быстрое кодирование	1955	Fort	1971
A-2, Flou-metik	1956	Prolog, Ci, Ada	1972
IPL-1, Mat-metik	1957	Smoltok	1980
Fortran	1958	VBA	1990
Алгол 58	1959	VC++	1993
APT, LISP, Kobol, Algol-60	1960	Java	1994
PL/1, Basic	1964	Delphi	1995
Алгол W	1965	C#	2000

Одним из самых распространенных на сегодня языков программирования считается **Паскаль** (Pascal). Язык программирования Паскаль был создан Николасом Виртом в 1969 году с целью обучения студентов программированию, но широко распространился среди профессиональных программистов.

Конечно, эффективные языки программирования не остаются неизмененными. И поэтому для разных типов компьютеров разрабатываются соответствующие версии языка Паскаль, и они заметно отличаются от начальной версии.

Вопросы и задания

1. *Что такое программа?*
2. *Что понимается под выражением «язык программирования»?*
3. *Чем отличаются разные уровни языков программирования?*
4. *Зависит ли язык программирования от типа компьютера? Поясните ответ.*
5. *Расскажите о нескольких языках программирования высокого уровня.*

Урок 13. Интегрированная среда Turbo Pascal 7.0

Широкое распространение языка Паскаль объясняется его простотой и удобством применения. Первоначально язык Паскаль применялся в университетах, но потом появились версии языка Паскаль для различных типов компьютеров. В 1981 году был предложен международный стандарт языка Паскаль. В настоящее время широко используется версия **Turbo Pascal 7.0**, разработанная фирмой **Borland**. Он предоставляет пользователям очень удобную систему — **интегрированную среду программирования**.

Интегрированная среда — это программа, поддерживающая программирование. Она должна отвечать следующим основным условиям:

- прежде всего, предоставление возможности ввода текста программы;
- сохранение текста программы время от времени на внешнем носителе;
- наличие транслятора для запуска программы;
- наличие средства для проверки и обнаружения синтаксических ошибок.

Интегрированная среда Турбо Паскаль 7.0, кроме перечисленного выше, предоставляет возможность выполнить еще и много других операций.

Система программирования «Турбо Паскаль 7.0» размещается обычно в каталоге «TP7» внешней памяти (можно размещать и в других каталогах). Она содержит более ста файлов, которые, в зависимости от назначения, хранятся в нескольких каталогах (рис. 1). Запускающий интегрированную

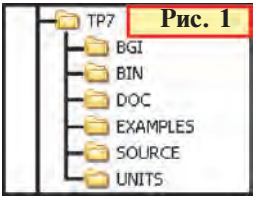


Рис. 1

среду Турбо Паскаль файл **Turbo.exe** 7.0. хранится в каталоге **«BIN»**. В остальных каталогах в основном находятся вспомогательные файлы, а также файлы, демонстрирующие возможности Турбо Паскаль.

Например, в каталоге **«BGI»** находятся файлы необходимые для работы в графическом режиме.

Об остальных файлах и каталогах можно получить информацию из дополнительной литературы.

После запуска файла Turbo.exe на экране появляется интерфейс интегрированной среды Турбо Паскаль. Он состоит из **строки меню, области окон и информационной строки** (рис. 2). В Турбо Паскаль различают несколько типов окон, и самое существенное для нас окно **текстового редактора программы.** Для его создания достаточно выбрать команду **New** (Новое) из меню **File**.

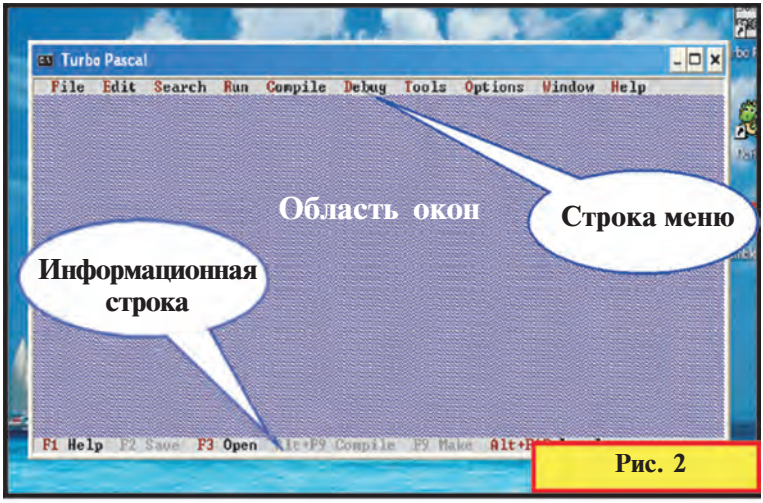


Рис. 2

Для перехода к строке меню нажимается клавиша F10. После этого с помощью клавиш со стрелками влево или вправо выбирается нужное меню и нажимается клавиша ENTER. Нужное меню можно выбрать и с помощью мышки.

Интерфейс интегрированной среды Турбо Паскаль очень схож с интерфейсом текстового редактора. В окне текстового редактора программы текст программы вводится так же, как в текстовом редакторе. Как и MS Word, в Турбо Паскале

можно открыть несколько окон, и с каждым из них можно работать отдельно. Это позволяет одновременно работать с несколькими программами.

Окно, в котором пользователь работает в текущее время, называется **активным окном**. В текстовом редакторе программы имя текста предлагается как **NONAME00.PAS,** что означает имя текста **безымянный00** и расширение файла **pas**.

Меню **File (Файл)** включает в себя следующие действия: Open (Открыть – загрузить файл из внешней памяти в оперативную память), Save (Сохранить), Save as (Сохранить под другим именем), Exit (Выход), меню **Edit (Правка)** — Cut (Вырезать); Copy (Копировать); Paste (Вставить), меню **Run** — команду запуска программы, меню **Compile** — команду компиляции программы (т.е. перевод программы на машинный язык и сохранение в виде файла с расширением «EXE»).

Действия, включенные в меню, можно выполнять и с помощью известных **(горячих)** клавиш. Ниже приведен перечень горячих клавиш, предназначенных для выполнения основных операций:

F3	Открыть	**Ctrl+F9**	**запуск программы**
F2	Сохранить	**Alt+F5**	просмотр результата программы
Alt+F3	Закрыть активное окно	**Alt+F9**	выполнить компиляцию программы
Alt + x	Выход	**F6**	Переход с одного окна к другому

При редактировании текста программы можно пользоваться следующими клавишами:

Клавиши направлений (←,→,↑,↓) — перемещение курсора в требуемом направлении;

Shift + (←,→,↑,↓) — выделение части текста программы в выбранном направлении, начиная с места расположения курсора;

Ctrl + Insert — копирование выделенной части текста программы в буфер-память;

Shift + Insert — вставка скопированной части из буфера в текст с места расположения курсора;

Shift + Delete — вырезание выделенной части текста.

Вопросы и задания

1. *В каком каталоге находится файл, запускающий Турбо Паскаль?*
2. *Что такое интегрированная среда программирования?*
3. *Откройте редактор текста программы в интегрированной среде Турбо Паскаль.*
4. *Что означает NONAME.PAS? В какой программе предлагается такое же имя?*
5. *Какая пара клавиш запускает программу, написанную в текстовом редакторе программы интегрированной среды Турбо Паскаль?*

Упражнения

Откройте окно редактора текста программы Паскаль и выполните следующие задания:

а) введите первый куплет гимна Республики Узбекистан;

б) откройте новое окно и введите второй куплет гимна Республики Узбекистан;

в) скопируйте текст со второго окна (второй куплет) и вставьте его в продолжение первого куплета на первом окне;

г) сохраните текст первого окна под именем «Gimn.txt»;

д) второе окно закройте без сохранения.

Урок 14. Алфавит языка Паскаль и его структура

Как и всякий язык программирования, Турбо Паскаль имеет свой алфавит и синтаксические правила.

Язык Турбо Паскаль включает набор символов с кодом ASCII, например:

26 заглавных и строчных букв латинского алфавита: Aa, Bb, Cc, Dd, Ee, Ff, Gg, Hh, Ii, Jj, Kk, Ll, Mm, Nn, Oo, Pp, Qq, Rr, Ss, Tt, Uu, Vv, Ww, Xx, Yy, Zz (для записи комментариев и текстов можно использовать также буквы кириллицы).

Десять арабских цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Шестнадцатеричные числа: арабские цифры от 0 до 9 и буквы A, B, C, D, E, F и a, b, c, d, e, f.

Специальные знаки: ″.″ (точка), ″,″ (запятая), ″:″ (двоеточие), ″;″ (точка с запятой), ″'″ (апостроф), ″"″ (кавычки), ″!″ (восклицательный знак), ″?″ (вопросительный знак), ″%″ (процент), ″$″ (доллар), ″@″ (коммерческий знак), ″&″ (амперсанд), ″#″ (решетка), ″^″ (обозначение

типа ссылки); различные скобки: (,), {, }, [,]; пары символов: :=, .. , (*, *), (., .).

Управляющие символы: с кодами от #0 до #31 (# — означает десятичное значение кода символа, во время работы управляющие символы на экране не отражаются).

На языке Турбо Паскаль, в основном, используются следующие операции и соответствующие им символы:

Арифметические операции: ″+″ (сложение), ″−″ (вычитание), ″*″ (умножение), ″/″ (деление).

Знаки операций сравнения: ″=″ (равно), ″<″ (меньше), ″>″ (больше); пары символов: ″<>″ (не равно), ″<=″ (не больше), ″>=″ (не меньше).

Логические операции:

AND (**«И» —** логическое умножение)	**OR** (**«ИЛИ» —** логическое сложение)
NOT (**«НЕ» —** логическое отрицание)	**XOR** (исключающее «ИЛИ»)

Как и всякий язык программирования, язык Паскаль имеет свою орфографию и правила, на основе которых с помощью вышеперечисленных букв, символов и действий составляются команды и указания. Каждое указание завершается знаком «;» (точка с запятой). В тексте программы одна строка содержит не более 127 символов.

Обычно, чтобы программа была понятной, в нее добавляют примечания. С помощью примечаний объясняют, какую задачу выполняет программа или ее части. В Турбо Паскаль примечание записывается между скобками вида { и } или (* и *).

Например, {это примечание написано для примера} или (*и так можно написать примечание *).

Обычно, программы на языке Паскаль начинаются со специального слова **Program**. После него записывается заголовок программы. Например:

Program kvadratnoye_uravneniye; {программа решения квадратного уравнения.}.

Заголовок программы должен отражать суть программы, что дает возможность отличить нужную программу среди других. Следует отметить, что заголовок программы никоим образом не влияет на работу программы и добавление

заглавия в программу считается необязательным. При составлении программ на языке Паскаль применяются следующие элементы:

Константы (постоянные) — величины, значения которых не меняются в процессе выполнения программы.

Переменные — величины, значения которых изменяются в процессе выполнения программы.

Выражения — константы, переменные и функции, связанные соответствующими операциями.

Операторы — команды языка программирования, заменяющие алгоритмические указания.

Функции и процедуры — отдельные части программы, имеющие собственное имя. Обращение к ним выполняется из основной программы.

Метки — указывают на оператора, которому передается управление.

Команды и указания языка Паскаль размещены в специальных файлах с расширением **TPU**, именуемым **модулем**. В качестве примера можно привести модули system (систем), **crt** (сиэрти) и **graph** (граф). Каждый из модулей содержит команды и указания определенного направления. Например, модуль **system** располагает набором стандартных (основных) команд языка Паскаль, в то время как команды модуля **crt** предназначены для работы с экраном и с клавиатурой (очистить экран, создать на экране несколько отдельных окон и т.д.), а модуль **graph** содержит команды и указания, предназначенные для работы в графическом режиме. При составлении программ модули используются по необходимости. Для использования команды из состава некоторого модуля в начале программы (после заголовка) необходимо ввести соответствующие указания. А это осуществляется с помощью специального слова **Uses** языка Паскаль. Например, для использования графики в программе необходимо ввести запись **Uses graph**;. Если в программе необходимо использовать несколько модулей, то их можно написать через запятую. Например, **Uses crt, graph**;

Модуль **system** подключается автоматически при загрузке интегрированной среды языка Турбо Паскаль. И поэтому запись **Uses system;** неуместна. Для работы большинства программ является достаточным модуль **system**.

При составлении программ сначала следует определить участвующие в ней величины, присвоить имена переменным и **описать** (указать) их тип. После этого начинается **основная часть** программы, т.е. на языке Паскаль программа состоит из двух частей.

В общем виде программа на языке Паскаль имеет следующую структуру:

Program имя программы; {необязательное}
Uses {список модулей}
Label {список меток}
Const {описание констант}
Var {описание переменных}
Процедуры и функции
Begin
{основная часть}
End.

Label, Const, Var, Begin, End — специальные слова языка Паскаль, которые означают: **label** — метка; const (constant — константа) — постоянная величина; **var** (variable — переменная) — описание переменных; **begin** — начало, **end** — конец.

Под словом **идентификатор** подразумевают имена констант, переменных, процедур, функций, модулей, программ. Идентификаторы делятся на **стандартные** и **пользовательские** типы.

Стандартные – это заранее определенные идентификаторы языком Паскаль.

Пользовательский идентификатор выбирается программистом и может иметь любую длину, **но значащими** (различающими) считаются первые 63 символа. Имя идентификатора должно начинаться с латинской буквы или со знака подчеркивания (_) и пишется без пробелов. После первого символа разрешается использовать латинские буквы, цифры и знак подчеркивания.

На языке Турбо Паскаль не различают регистр (нижний или верхний) записи имен идентификаторов, т.е. имена **aka**, **Aka**, **aKa** считаются идентичными **именами**, потому что транслятор Турбо Паскаль в процессе компиляции программы (перевода программы на машинный язык) все заглавные буквы имен идентификаторов и специальных слов

переводит в строчные буквы. Имена не записываются внутри апострофа, т.е. **'Men'** и **'men'** не считаются именами.

Следующие ключевые слова зарезервированы языком Турбо Паскаль, и поэтому не могут быть использованы как пользовательский идентификатор:

and, asm, array, begin, case, const, constructor, destructor, div, do, downto, else, end, exports, file, for, function, goto, if, implementation, in, absolute, assembler, export, external, far, forward, index, interrupt, near, private, public, resident, virtual, inherited, inline, interface, label, library, mod, nil, not, object, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor.

Язык Турбо Паскаль в программе не разрешает использовать эти слова и указывает на ошибку сообщением **«Error 2: Identifier expected»** (зарезервированный идентификатор).

Вопросы и задания

1. *Расскажите об алфавите языка программирования Паскаль.*
2. *Объясните логические операции с помощью таблицы истинности.*
3. *Что такое оператор?*
4. *Расскажите о заголовке программы.*
5. *Расскажите об идентификаторе.*
6. *Из каких частей состоит программа на языке Паскаль?*
7. *Расскажите о части описания программы.*

Упражнения

1. Определите соответствующие символы правого столбца к множествам левого столбца.

Логические операции	**%, $, @ , & , (,), {, }, [,]**
Знаки операций сравнения	NOT, OR
Специальные знаки	<, <=, >, >=

2. Определите соответствующие фразы левого столбца к описаниям правого столбца.

Использование модулей	Program 9_sinf;
Описание меток	Var a21: integer;
Заглавие программы	Label 19;
Описание переменной	Uses Crt;

3. Разделите имена идентификаторов на группы «Правильная запись» и «Неправильная запись». Объясните ответ.

a	1kun	Mening Birinchi Dasturim	BMA
Chegara#4	Keyingi yil	Kun_21_iyul_1963	and

Урок 15. Постоянные и переменные величины

На языке Паскаль, в основном, используется три вида величины: **постоянные, переменные** и **в виде таблицы**. Они могут принимать символьные, строковые, логические или числовые значения.

Постоянные величины

Символьная константа — заключенный в апострофы один символ: буква, цифра или специальный знак. Например, 'a'; 'B'; '9'; '-', ' ' и т.д.

Строковая константа (строка, состоящая из символов) — последовательность букв, цифр и специальных знаков (кроме символа с ASCII кодом 13), заключенная в апострофы и с количеством символов от 0 по 122 символов. Например, 'Tashkent'; 'A 549'; ' '; '47%'; 'BMA ='; '..-...-' и т.д. Если в строке нужно указать сам символ апострофа, он удваивается.

Если между апострофами пусто, она называется **пустая строка**.

Логическая константа принимает одно из двух значений True (истина) или False (ложь).

Числовые константы могут быть двух типов: **целые** или **вещественные**. **Целые числа** записываются со знаком или без него по обычным правилам арифметики и могут иметь значение от − 2147483648 до +2147483647. Следует учесть, что, если целочисленная константа выходит за указанные границы, компилятор дает сообщение об ошибке. Вещественные числа, в свою очередь, делятся на числа с фиксированной точкой и плавающей точкой.

На языке Паскаль для отделения целой части числа от десятичной дроби вместо «запятой» пишется «точка».

Числа с фиксированной точкой — это числа в виде десятичной дроби. Например:

− 2.753; 283.45; 0.517; − 0.0013.

Числа с плавающей точкой — числа, отображаемые в экспоненциальном (с помощью E или e) виде. Этот метод удобен при записи очень маленьких или очень больших чисел. Они читаются так:

2.1E+07 — «2.1 умноженное на 10 в степени 7»;

2.301e−63 — «2.301 умноженное на 10 в степени минус 63».

Например, число 3400000000 = $3{,}4 \cdot 10^9$ на языке Паскаль имеет экспоненциальный вид как 3.4E+09. Число перед буквой **E** называют **мантиссой**, а число за буквой **E**, называется **порядком**. Мантисса может быть и целым, и дробным числом, а порядок − только целым. Например:

37.3879 E−3= 0.0373879; 5.31 E+5= 531000; −0.075 E−5= = −0.00000075; −2.37 E−4= − 0.000237.

В программу, составленную на языке Паскаль, можно включить **объявляемые константы**. Например:

Const A=21071963; _m10m10='2301'; Pi=3.141516;

Переменные величины

Переменные обязательно должны быть описаны в части описаний программы, т.е. следует указать тип переменной. Описание переменных на языке Паскаль начинается со специального слова **Var**:

Var переменная: тип;
переменная: тип;

В случае с несколькими переменными одного и того же типа, можно обойтись не отдельным, а общим описанием:

Var 1-переменная, 2-переменная, …, n-переменная: тип;

Переменные, значениями которых могут быть только целые числа, называются **целочисленными переменными.** Они разделяются на 5 типов, которые отличаются диапазоном возможных значений и занимаемым местом (объемом) в памяти компьютера.

В следующей таблице приведены специальные слова языка Паскаль для описания переменных, диапазон возможных значений и занимаемый объем в памяти:

Тип	Диапазон значений	Занимаемый объем в памяти
Byte	0 …255	8 битов = 1 байту
ShortInt	−128 …127	8 битов = 1 байту

ShortInt	−128 ...127	8 битов = 1 байту
Word	0 ...65 535	16 битов = 2 байтам
Integer	−32 768 ...32 767	16 битов = 2 байтам
LongInt	−2 147 483 648 ...2 147 483 647	32 битов = 4 байтам

Например: var i, j: Integer; bma: longint; mmr: Shortint;
nomer: Byte; nat_0: word;

Над целыми числами имеют места действия **div** (деленые в целом) и **mod** (остаток). Например:

25 div 4 = 6; 25 mod 4 = 1; 49 div 7 = 7; 49 mod 7 = 0.

Переменные, значениями которых могут быть вещественные числа, называются **вещественными переменными**. В следующей таблице приведены информации о них:

Тип	Диапазон значений	Разряд	Занимаемый объем в памяти
Real	$-2{,}9 \cdot 10^{39}$ ...$1{,}7 \cdot 10^{38}$	11—12	6 байтов
Single	$-1{,}5 \cdot 10^{45}$ ...$3{,}4 \cdot 10^{38}$	7—8	4 байта
Double	$-5{,}0 \cdot 10^{324}$ ...$1{,}7 \cdot 10^{308}$	15—16	8 байтов
Extended	$-3{,}4 \cdot 10^{4932}$ ...$1{,}1 \cdot 10^{4932}$	19—20	10 байтов
Comp	$-9{,}2 \cdot 10^{18}$ ...$9{,}2 \cdot 10^{18}$	19—20	8 байтов

Например:

var Var ugol, dlina_dugi : Real;: Real; mab : extended;
stepen : Single; kub : double; veshestvenniy : Comp;

Разряд в таблице означает количество точных цифр числа. В большинстве случаев будет достаточно пользоваться переменной типа **real**. Переменные, значениями которых являются символьные константы, называются **символьными переменными**. Для их описания используется специальное слово языка Паскаль **Char**. Например, Var bukva, simvol: char;

Для описания **строчных переменных** используется специальное слово языка Паскаль: **String**. Для этих переменных в памяти компьютера отводится 255 байтов (каждому знаку по байту). Если известно, что длина строковой переменной в процессе работы программы не превысит конкретного значения, например не превысит 10-ти знаков, то в целях экономии памяти можно описать ее следующим образом: String [10]. Например:

var stroka : String; {переменной stroka выделено из памяти 255 байтов}

_stroka: String[24]; {переменной _stroka выделено из памяти 24 байта}

Логические переменные описываются специальным словом языка Паскаль: Boolean.

Например:

```
var   rezultat : Boolean;
      bolshe, menshe: Boolean;
```

В программе, составленной на языке Паскаль, разрешается использовать только те переменные, которые были описаны. Транслятор языка Паскаль в программе не разрешает использовать не описанные переменные и указывает на ошибку сообщением **«Error 3: Unknown identifier»** (неизвестный идентификатор, т.е. в этом случае неизвестная переменная). Следует помнить, что переменные могут принимать значения, указанные в описании типов.

Вопросы и задания

1. *Что понимается под символьной величиной? Приведите примеры.*
2. *Чем отличаются строчные и символьные константы?*
3. *Какие типы числовых констант вы знаете?*
4. *Какие значения могут принимать логические константы?*
5. *Чем отличаются переменные и константы?*
6. *Приведите примеры типов целых переменных.*
7. *Приведите примеры типов вещественных переменных.*
8. *Как описываются символьные переменные? Приведите примеры.*
9. *Как описываются строковые переменные? Приведите примеры.*

Упражнения

1. Расскажите о типах следующих констант.
   а) '7!'; 'informatika'; '−987378'; 'ВМА';
   б) ','; 'u'; '0'; ' '; в) 99; −200; 101; 87;
   г) 0.01; 8.909; 132.001; 878887.1;
   д) 0.07 E−3; −9.8 E6; е) True; False.
2. Определите тип следующих переменных и дайте пояснение.
   а) men : Boolean; б) bahodir : String[7];
   в) hayot : Real; г) son : char;
   д) baxt : Integer; е) ser : Single;

Урок 16. Повторение темы «Постоянные и переменные величины»

1. Расскажите о типах следующих констант.
   а) −9.22 E−2; 0.01 E+5; 1.11 E−4;
   б) 21; 21; 7; 7; 19; 19; 63; 63;
   в) true; true; true; false; false; false;
   г) '555'; 'aar'; 'mmr'; 'bbj'; 'aga';
   д) 'Muqaddas'; 'Vatan';'Mustaqil';
   е) 'i'; 'n'; 's'; 'o'; 'n'; 'i'; 'y'; 'a'; 't';
2. Опишите следующие переменные, присвоив им имена.
   а) символьная; б) вещественная;
   в) логическая; г) строковая;
   д) строковая длиной не более 7 символов.
3. В каждом пункте указаны все значения или отражены свойства переменных. Опишите эти переменные, присвоив им имена.
   а) −5; 0; 7; 58; −15; 9; б) 'Xalq'; 'Vatan'; 'Ona';
   в) 7.21; 4.2; 50.1902; −1.23; г) первые 7 простые числа;
   д) true; true; false; true; false; е) '000'; '001'; '002'; '003';
   ж) буквы алфавита;
   з) 'Yuksak'; 'ma'naviyat', 'yengilmas'; 'kuch'.
4. Указаны все значения переменных. Опишите целые переменные, присвоив им имена. Выберите тип переменных так, чтобы они занимали самый меньший объем в памяти.
   а) −4; 0; −4; 8; 12; б) 1; 16; 256; 4096; 65536;
   в) 0; 2; 4; 6; 8; 10; г) 29350; −2; 8000; 250;
   д) 5; −32767; 46; 0; 32767; е) 200000; 2000; −20; 99999;

5*. Опишите заданные переменные и переменные, получаемые в результате действий.
   а) f: целое; g: вещественное; d:=f*g+f+g;
   б) d: целое; n: целое; k:=d+2*n;
   в) s: логический; e: логический; q:= not(s or e);
   г) k: нечетное; m: четное; vvv:= k+m/2;

Урок 17. Табличные величины

В повседневной жизни мы сталкиваемся с самыми разнообразными таблицами: расписание уроков, таблица шахматных или футбольных соревнований; таблица (умножения) Пифагора, таблица падежей и много других. Составляющие содержания таблиц называются ее **элементами**.

Табличные величины бывают **одномерные** (линейные), **двумерные** (прямоугольные), **трехмерные** (параллелепипед-

ные) и т. д. В учебнике будут рассмотрены линейные и прямоугольные таблицы.

Линейные таблицы выражаются в виде строки или столбца. Например, список учеников в классном журнале имеет вид таблицы в форме столбца. Фамилии учащихся являются элементами этой таблицы. Каждому из них соответствует свой порядковый номер, и каждому номеру соответствует фамилия только одного ученика.

Двумерные таблицы состоят из строк и столбцов (вспомните текстовый процессор и электронную таблицу). Элементы этих таблиц размещаются на пересечении строк и столбцов. Чтобы указать какой либо элемент двумерной таблицы, нужно знать номер строки и номер столбца, на пересечении которых находится этот элемент. Следовательно, каждому элементу двумерной таблицы соответствует два порядковых номера (номер строки и номер столбца).

На языке Паскаль для работы с таблицами введено понятие **массива**. **Массив** — это табличная величина, имеющая фиксированное число пронумерованных элементов (с порядковыми номерами) одного типа. Порядковые номера элементов массива выражаются в целых числах, т.е. они могут быть и **отрицательными**.

На языке Паскаль каждый массив должен иметь свое имя, с ограничениями как у переменных.

Например, a5, tablitsa_urokov, prostiye_chisla.

Порядковые номера элементов массива называются **индексом** и записываются в квадратных скобках.

Например, запись a[5] означает пятый элемент массива a, т.е. имя массива — a, индекс — 5.

Пример 1. Составьте линейную таблицу A с 7 элементами.

Порядковый номер	1	2	3	4	5	6	7
Значение	x	A	a	t	r	z	m

Следовательно, элементы таблицы и их значения имеют следующее соответствие:

Порядковый номер	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
Значение	x	A	a	t	r	z	m

Элементы двумерного массива определяются через два индекса. Индексы записываются через запятую, и первый индекс определяет номер строки, а второй – номер столбца.

Например, запись S[4,3] означает элемент массива S, расположенный на пересечении 4 строки 3 столбца.

Пример 2. Опишите прямоугольную таблицу с именем S и размером 4×5 (4 на 5, т.е. 4 строк и 5 столбцов; элементы таблицы вписаны в ячейки синим цветом).

Порядковые номера строк	Порядковые номера столбцов				
	2	3	4	5	6
1	3.2 S[1,2]	1.37 S[1,3]	−1.25 S[1,4]	7.12 S[1,5]	−11.4 S[1,6]
2	0.5 S[2,2]	1.1 S[2,3]	1.2 S[2,4]	−1,1 S[2,5]	4.22 S[2,6]
3	−0.1 S[3,2]	1.01 S[3,3]	71.2 S[3,4]	4.1 S[3,5]	−4.11 S[3,6]
4	6.3 S[4,2]	−7.01 S[4,3]	1.5 S[4,4]	7.5 S[4,5]	−1.09 S[4,6]

Из таблицы видно, например, [1,3]=1.37, S[2,2]=0.5, S[4,6]= = −1.09.

В программе массивы должны быть описаны так же, как и переменные. Для описания служит зарезервированное слово языка Паскаль **Array.** За словом Array в квадратных скобках записываются номера первого и последнего элементов, разделенные **двумя точками** (..). А затем записывается зарезервированное слово **of** и тип элементов массива.

Например:

```
var
A: array [1..7] of char; {линейный массив с именем A типа char (с символьными значениями), из примера 1, с упорядоченными от 1 по 7 элементами};
S: array [1..4, 2..6] of real; {двумерный массив с именем S типа real (с вещественными значениями), из примера 2, с упорядоченными элементами по строке от 1 по 4 и по столбцу от 2 по 6};
bma: array [−2..100] of integer; {упорядоченный от −2 по 100 линейный массив целого типа с именем bma}.
```

Следовательно, под **массивом** (табличной величиной) понимается **совокупность упорядоченных величин одного типа, объединенных одним именем**.

Пример 3. Пусть одномерная таблица A имеет 5 элементов:

Порядковый номер	−1	0	1	2	3
Значение	3	2	12	10	−8

На языке Паскаль элементы массива записываются следующим образом:

A[−1] := 3; A[0] := 2; A[1] := 12; A[2] := 10; A[3] := −8;

Индекс элементов массива можно обозначить целочисленным переменным (например, i), например, если i = 1, то A[i] = −12, если i = 3, то A[i] = −8.

Пример 4. Пусть задан двумерный целочисленный массив B:

$$B = \begin{bmatrix} 3 & 10 & 5 \\ 2 & 7 & 9 \end{bmatrix}.$$

Зададим порядковые номера элементам массива и перепишем как B[0,0], B[0,1], B[0,2], B[1,0], ...:

$$B = \begin{bmatrix} 3 & 10 & 5 \\ 2 & 7 & 9 \end{bmatrix} = \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \end{bmatrix} = \left[B_{ij} \right],$$

здесь i = 0, 1 и j = 0, 1, 2 (i — порядковые номера строк, j — порядковые номера столбцов). Эта таблица на языке Паскаль описывается следующим образом:

```
var  b: array[0..1, 0..2] of Integer;
```

Напоминаем, что тип регистра записи имен идентификаторов не имеет значения!

В общем случае вместо индекса используются переменное или выражение.

Например, если $I = 0$, $J = 2$, то в примере 4: B[I, J] = 5 и B[I+1, J−2] = 2 (так как I+1 = 0+1 = 1 и j−2 = 2−2 = 0).

Вы познакомились только с линейными и прямоугольными видами таблиц. В общем случае на языке Паскаль можно использовать многомерные (по 255) табличные величины. Приведем несколько примеров описания таких таблиц:

1) var s: array[1..4, 1..7, 0..10] of Byte; {s − 3 мерная таблица типа Byte};

2) var t, k: array [1..100, 1..80, 1..50] of string; {t и k − 3 мерные таблицы строчного типа};

3) var f: array [−5..10, 0..10, 2..10] of char; {f − 3 мерная таблица символьного типа};

На языке Паскаль для описанных массивов в памяти резервируются места. И поэтому, чтобы не занимать лишние места в памяти кроме типа массива, желательно знать количество элементов массива.

В общем случае, количество элементов линейного массива с упорядоченными от K до S элементами равно **S−K+1**, количество элементов двумерного массива с упорядоченными элементами по строке от B по M и по столбцу от A по G равно **(M−B+1) · (G—A+1)**.

Например, у массива A из примера 3, упорядоченного от −1 по 3, 5 целочисленных элементов (3−(−1)+1 = 3+1+1 = 5), у массива B из примера 4, упорядоченного по строке от 0 по 1 и по столбцу от 0 по 2, 6 целочисленных элементов ((1−0+1)·(2−0+1) = 2 ·3 = 6).

Вопросы и задания

1. *Приведите примеры таблиц, часто встречающихся в повседневной жизни.*
2. *Какой размер имеет линейный массив?*
3. *Для чего служит индекс в массиве?*
4. *Какие типы значений могут принимать индексы массива?*
5. *Как можно определить тип элементов табличной величины?*

Упражнения

1. Определите размерность и количество элементов массивов, заданных следующими последовательностями:

а) A[0], A[1], A[2], A[3], ... , A[99];

б) B[0,0], B[0,1], B[0,2], ... , B[3,5];

в) M[0,0,0], M[0,0,1], ... , M[1,1,1];

г) G[−22,3], G[−22,4], G[−22,5],..., G[−20,5].

2. Укажите правильное описание целочисленного линейного массива из 100 элементов.

а) var B: array [1..100] of real;

б) var M: array [1..100] of char;

в) var A: array [0..99] of string;

г) var G: array [5..104] of integer;

3. Определите тип, размерность и количество элементов массивов из примера 2.

4. Опишите линейную целочисленную и символьную, а также двумерную вещественную таблицы.

Урок 18. Повторение темы «Табличные величины»

1. Определите описание прямоугольной вещественной таблицы F с 8 строками и 11 столбцами.

а) var A: array [8..11] of real;

б) var B: array [1..8,1..11] of integer;

в) var D: array [8..11,8..11] of real;
г) var M: array [0..8,0..10] of integer;
д) var F: array [0..7,0..10] of real;
е) var F: array [0..7,0..10] of char;

2. Определите тип, размерность и количество элементов массивов из примера 1.

3. Составьте таблицу Пифагора и проанализируйте ее элементы. Дайте описание массива, присвоив ему имя.

4. Составьте таблицу, состоящую из имени, даты рождения и образования членов вашей семьи. Дайте описание массива и проанализируйте его элементы.

5. Введите в пустые ячейки таблицы значения, соответствующие элементам массива M типа **integer**. Дайте описание массива.

M[−7]	M[−6]	M[−5]	M[−4]	M[−3]	M[−2]	M[−1]

6. Введите в пустые ячейки таблицы значения, соответствующие элементам массива B типа **char**. Дайте описание массива и перепишите элементы массива в виде столбца. Переведите массив в двумерный вид и заново опишите.

B[9]	B[10]	B[11]	B[12]	B[13]	B[14]	B[15]	B[16]

Урок 19. Стандартные функции и процедуры, алгебраические выражения

Понятие функции вам известно из курса математики. Функции, по присущим им свойствам, разделяют на семейства. Например, линейные, квадратичные, тригонометрические и т.д. Некоторые функции из упомянутых семейств используются на языке Паскаль. Они предопределены транслятором языка Паскаль и называются **стандартными функциями.**

Эти функции вам знакомы через программы Excel. На языке Паскаль, наряду со стандартными функциями, используются **стандартные процедуры**, выполняющие определенные действия.

Ниже приведены некоторые стандартные функции и процедуры языка Паскаль:

Функ-ция	Тип аргумента	Тип значения	Комментарий
Математические функции			
Abs(x)	целый/ вещественный	целый/ вещественный	Абсолютное значение (модуль) **x**: **|x|**
Sin(x)	целый/ вещественный	вещественный	синус **x** (аргумент x в радианах): **sinx**
Cos(x)	целый/ вещественный	вещественный	косинус **x** (аргумент x в радианах): **cosx**
Arctan (x)	целый/ вещественный	вещественный	арктангенс x (аргумент x в радианах): **arctgx**
Sqrt(x)	целый/ вещественный	вещественный	квадратный корень x **(x ≥ 0):** $\sqrt{x}$
Sqr(x)	целый/ вещественный	целый/ вещественный	квадрат аргумента **x: x^2**
Exp(x)	целый/ вещественный	вещественный	**e^x** (e = 2.718282...)
Ln(x)	целый/ вещественный	вещественный	натуральный логарифм **x (x > 0): ln x**
Frac(x)	целый/ вещественный	вещественный	дробная часть **x: {x}**
Int(x)	целый/ вещественный	вещественный	целая часть аргумента **x: [x]**
Random	—	вещественный	случайное число в промежутке **[0, 1)**
Random (x)	Word	Word	случайное число в промежутке **[0, x)**
Функции преобразования типов переменных			
Trunc (x)	вещественный	LongInt	целая часть **x**
Round (x)	вещественный	LongInt	округляет значение **x** до целого
Odd (x)	целый	логический	если **x** нечетное, то принимает значение «истина»
Chr (x)	Byte	Char	символ, соответствующий десятичному коду **x** в ASCII
Ord ('m')	Char	Byte	десятичный код ASCII символа **'m'**
Математические процедуры			
Inc (x)	целый	целый	увеличить **x** на единицу **(x:=x+1)**
Dec (x)	целый	целый	уменьшить **x** на единицу **(x:=x−1)**

Пример 1. Применение некоторых функций.

Функция	Значение	Функция	Значение	Функция	Значение
abs(−5)	5	abs(−4.9)	4.9000000000e+00	abs(4.9)	4.9000000000e+00
sqr(4)	16	sqr(2.5)	6.2500000000e+00	Sqrt(16)	4.0000000000e+00
sqr(−4)	16	Sqr(0.0)	0.0000000000e+00	Sqrt(0.16)	4.0000000000e−01
sqr(0)	0	Sin(0)	0.0000000000e+00	Sin(1)	8.4147098481e−01
trunc(5.3)	5	Int(5.3)	5.0000000000e+00	Int(5)	5.0000000000e+00
trunc(−5.3)	−5	Int(−5.3)	−5.0000000000e−00	frac(5.3)	3.0000000000e−01
Round(5.49)	5	frac(−5.3)	−3.0000000000e−01	frac(5)	0.0000000000e+00
Round(5.5)	6	Odd(5)	TRUE	Odd(−5)	TRUE
Round(−5.49)	−5	Odd(4)	FALSE	Odd(−4)	FALSE
Round(−5.5)	−6	Odd(0)	FALSE	Chr(65)	'A'
Chr(97)	'a'	Ord('A')	65	Ord('a')	97

Для обозначения числа π, используемого во многих математических формулах, на языке Паскаль введена специальная постоянная (константа) **Pi** (Pi=3,1415...).

На языке Паскаль **алгебраические выражения** составляются с помощью констант, переменных и функций, связанных арифметическими действиями. Алгебраические выражения записываются на одной строчке, т.е. невозможно подстрочная или надстрочная запись. Например, выражение $3ab^2$ на языке Паскаль записывается в виде **3*a*sqr(b)** или **3*a*b*b**, а выражение $\frac{a}{b^2}$ в виде **a/sqr(b)** или **a/(b*b)**.

Чтобы **указать** порядок выполнения действий, при записи выражений используются только обычные скобки. Выполнение действий внутри скобок такое же, как принято в математике – слева направо:

- вычисляются значения функций;
- выполняется умножение или деление;
- выполняется сложение или вычитание.

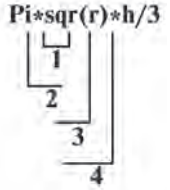
Например, выражение $\frac{a+b}{c}$ на языке Паскаль записывается в виде **(*a*+*b*)/*c*** и при этом сначала вычисляется выражение в скобках, т.е. *a*+*b,* затем результат делится на *c*. Порядок выполнения действий

упорядочивается скобками: выражение $\sqrt{a^2 - b^2}$ на языке Паскаль записывается в виде **sqrt(sqr(a) − sqr(b))**, а выражение |x + tgx| в виде **abs(x + sin(x)/cos(x))**.

Пример 2. При заданных значениях переменных R и H требуется вычислить значение выражения:

$$\frac{1}{3}\pi R^2 H.$$

Это выражение на языке Паскаль записывается в виде **Pi*sqr(r)*h/3.** При этом действия выполняются в следующем порядке:



1. **sqr(r)**
2. **Pi*sqr(r)**
3. **Pi*sqr(r)*h**
4. **Pi*sqr(r)*h/3**

Разрешается писать подряд идущие арифметические действия с помощью скобок. Например: 5*(−1) или a+(−b).

Иногда требуется записывать выражения, написанные на языке Паскаль в обычном математическом виде. Например, написанное на языке Паскаль выражение **0.5*(sin**(x)+**cos**(x)) в математическом виде выглядит так:

$$\frac{1}{2}(sinx + cosx).$$

Не все математические действия и функции нашли свое отражение в стандартных функциях языка Паскаль. Поэтому при преобразовании некоторых математических действий на язык Паскаль приходится использовать несколько стандартных функций или же одну стандартную функцию использовать несколько раз.

Например, на языке Паскаль нет функции возведения числа в произвольную степень. Поэтому, выражение a^3 на языке Паскаль принимает вид **a*a*a** или **sqr(a)*a**, а выражение a^4 будет выглядеть как **a*a*a*a** или **sqr(sqr(a))** или **sqr(a)*sqr(a)**.

В целом, для выражения a^b (a>0) в математике имеет место формула $\boldsymbol{a^b = e^{b \cdot ln a}}$. И поэтому выражение a^b (a>0) на языке Паскаль записывается в виде **exp(b*ln(a))**.

Пример 3. Написать алгебраическое выражение $\frac{x - y}{x^5 - y^3}$ на языке Паскаль.

Решение. Это выражение на языке Паскаль можно написать несколькими способами, одно из которых приводится ниже:

(x−y)/(exp(5*ln(x))−sqr(y)*y).

Следует помнить, что аргументы стандартных функций языка Паскаль записываются в скобке!

Вопросы и задания

1. *Какие функции называются стандартными?*
2. *Чем отличается обычная запись стандартных функций от записи на языке Паскаль?*
3. *Из чего состоит алгебраическое выражение?*
4. *Если в выражении встречаются несколько одинаковых действий, то в каком порядке они выполняются?*
5. *Чем можно изменить данный порядок математических действий?*
6. *Верно ли равенство Trunc(4.7)=Round(4.7)? Объясните ответ.*
7. *Почему выражение вида sin x−c на языке Паскаль считается ошибочным?*
8. *Соответствует ли требованиям языка Паскаль запись 2*− v? Объясните ответ.*
9. *Объясните порядок выполнения действий в выражении sqr(abs(x +sin(x))−pi).*

Упражнения

1. Переведите числа с плавающей точкой из примера 1 в числа с фиксированной точкой.

2. Напишите следующие алгебраические выражения на языке Паскаль:

а) $ax+b$; б) $xyz-23$; в) ax^2+bx+c;

г) $a^4x^3-(1-y^2)^2$; д) $\frac{a+5}{2b}$; е) $8(a+b^2c)$.

3. Напишите следующие выражения на языке Паскаль:

а) $25^{20}+|1-y^2|$; б) [5m]+{100b};

в) x sina+y cosb−5^2; г) sinsin x+cossin y;

д) $21-\sqrt{2011-x^2}$; е) $\sqrt[5]{x^3+a}-\sqrt{2}$.

4. Найдите среди следующих выражений на языке Паскаль записанные не правильно.

а) 2*a+b; б) sqr(x*b^2); в) sin(−3*x);

г) sin((a+b+cos(x)); д) 2*(−b)+a2.

5. Переведите следующие выражения, записанные на языке Паскаль, в обычный вид:

а) a*(Sqr(x)+1);

б) sin (x*x*x−sqr(sqr(x))+5);

в) pi*h*(sqr(r1) + sqr(r2) + r1*r2)/3;

Урок 20. Повторение темы «Стандартные функции и процедуры, алгебраические выражения»

1. Напишите следующие выражения на языке Паскаль:

а) $\frac{x-y}{x^2-y^3}$; б) $\frac{x+y}{xyz}+\sin^2 x$; в) $(5a^2+2x)+\frac{3x}{a^3}+tg^5a^3$;

г) $\cos^3\sin^2 x+\cos a^5$; д) $\sqrt{5+x}-\sqrt{z}\,\frac{3x}{a^3}+\sqrt[7]{a}$.

2. Вычислить значения следующих выражений, записанных на языке Паскаль:

а) sqr(trunc(4.95)); б) trunc(int(4.95)+0.7);
в) round(trunc(3.5)+0.7); г) 3+frac(12.5);
д) sqrt(sqr(16)); е) sqrt(sqrt(256)+9);
ж) sqr(5−abs(−5)); з) abs(−sqrt(16)).

3. При a=5 и b=4 вычислить значения следующих выражений, записанных на языке Паскаль:

а) abs(a+b−a*b); б) sqr(a+b−a*b)−110;
в) round(a/b+0.3)+9; г) 3+frac(b/a);
д) sqrt(sqr(a)−b*b); е) sqrt(sqrt(a+b)+6);
ж) sqr(a−abs(b−a)); з) abs(9−sqrt(a*b+a)).

4. Определить тип следующих констант, записанных на языке Паскаль:

а) abs(−sqrt(2011)); б) abs(sqr(2))+19;
в) frac(abs(−20)); г) int(1.9)*trunc(0.2).

Урок 21. Операторы присваивания и вывода информации на экран

Язык Паскаль, обычно для описанных переменных выделяет места из памяти и присваивает начальные значения, соответствующие типу.

Тип переменной	**Начальное значение**	**Тип переменной**	**Начальное значение**
все целочисленные	0	все вещественные	0.0000000000e+00
char	' ' (пробел)	boolean	FALSE
string	'' (пустая строка)	string[7]	'' (пустая строка)

Оператор присваивания. Оператор присваивания предназначен для присваивания значений переменным. Он выражается символом :=.

Общий вид оператора присваивания следующий:

переменная := выражение;

При задействовании этого оператора выполняется следующее:

1) вычисляется выражение;

2) результат присваивается переменной, т.е. значение выражения записывается на выделенное место («старое» значение переменной стирается) из памяти.

Пример 1. После выполнения следующей программы значение переменной **a** равно числу **22**.

```
var  a: integer;
begin
  a := 22;
End.
```

Пример 2. После выполнения следующей программы значение строковой переменной **«frukt»** равно слову **«olma»**.

```
var  frukt : string;
begin
  frukt := 'olma';
End.
```

Пример 3. В этом примере наглядно видно изменение значений переменных ***a*** и ***b***.

```
 var     a,b,m: integer;
 begin
a := 8;          {значение a равно  8}
b := a*5;        {значение b равно a*5=8*5= 40}
b := b+10;       {теперь значение b равно b+10= 40+10= 50}
m:=m*b;          {так как начальное значение m не задано, ей
                  присваивается значение 0, следовательно, значение
                 m  равно  0*50=0}
End.
```

В приведенных примерах переменным присваивались различные значения. Так как значения переменных остались в памяти компьютера, мы не видели эти значения. Потому что значения переменных на экран не выводились. Для вывода данных на экран используется **оператор вывода**. На языке Паскаль оператор (процедура) вывода имеет следующие два вида:

Write(список данных) и **WriteLn(список данных)**

здесь **Write** (англ. — писать) и **WriteLn** служебные слова, список данных — последовательность выражений, переменных или констант, разделенных между собой запятыми и намеченных для вывода на экран. Если в списке участвует выражение, то сначала производится вычисление, и полученный результат выводится на экран.

Если в списке данных участвуют символьные или строковые константы, то они обязательно должны заключаться в апостроф.

Отличие операторов **Write** и **WriteLn** состоит в том, что после задействования оператора **Write** курсор остается на текущей строке, т.е. следующие выводимые данные выводятся на эту же строку с позиции курсора. А после выполнения оператора **WriteLn** курсор переходит на начало следующей строки.

Пример 4.

```
begin
   write('Yashna, '); write('gulla ');
   write('ona Vatanim!');
End.
```

После выполнения программы на экране компьютера отражается запись

Yashna, gulla ona Vatanim!

Пример 5.

```
begin
     writeln('Yashna, ');
     writeln('gulla ');
   write('ona Vatanim!');
End.
```

После выполнения программы на экране компьютера отражается запись

Yashna,
gulla
ona Vatanim!

Пример 6.

```
program operator_prisvoyeniya;
var a,b:integer;
begin   a:=23; b:=a+21;
write('znacheniye b raven k ', b, '
(chislo)');
End.
```

После выполнения программы на экране компьютера отражается запись

znacheniye b raven k 44
(chislo)

При выводе данных можно указать **формат вывода**. Формат вывода определяет вид (формат) выводимых данных. Для этого после имени переменной вводится «:» (двоеточие). Например, если переменная a вещественного типа, то в формате вывода указываются два параметра — количество позиций, выделенных для выводимого числа. Например, оператор **WriteLn(a:10:2)**; для вывода значения *a* отводят 10 позиций (включая отрицательный знак и точку), из них выделяется одна позиция для точки и две позиции отводятся для вывода цифр дробной части. Если переменная целого типа, то в формате вывода задается количество позиций. Например, **WriteLn(b:6)**;. При выводе значений символьных и строчных переменных формат вывода определяет число позиций для вывода значения переменной (текста).

При выводе число или текст выравниваются по правому краю отведенного поля, например: если *a*: = 3.24;, то оператор WriteLn('a=', a:6:2); выводит на экран: *a* = 3.24 (после знака «=» оставляются две пустые позиции (пробелы)). При недостатке количества позиций для вывода число или текст выводятся полностью, а формат кроме формата дробной части числа игнорируется. Если формат вывода не задавать, то значения целых и строковых переменных выводятся полнос-

тью, а вещественных — в экспоненциальной форме с количеством цифр, соответствующих типу переменной.

Пример 7.

```
var a,b : real;
Begin a:=3.24; b:=5;
   writeln('a=',a); writeln('b=',b);
End.
```

На экране компьютера

```
a=3.2400000000E+00
b=5.0000000000E+00
```

Пример 8.

```
var a,b : real;
Begin  a:=3.24; b:=5.3;
writeln('a=', a:6:2);
writeln('b=',b:1:0);
End.
```

На экране компьютера

```
a=  3.24
b=5
```

В обоих приведенных примерах значения переменных *a* и *b* одинаковы, но вид сильно отличается. Выводимые на экран данные в примере 8 имеют вид яснее и понятнее, чем в примере 7. Чтобы получить более точные результаты следует осторожно пользоваться форматом вывода. Помните, для просмотра результатов программы нажимаются горячие клавиши **ALT+F5**.

Вопросы и задания

1. *Какие задачи выполняет оператор присваивания?*
2. *Объясните на примерах общий вид оператора присваивания.*
3. *Объясните на примерах общий вид оператора вывода.*
4. *Чем отличаются операторы Write и WriteLn?*
5. *Объясните на примерах возможности оператора вывода.*
6. *Что означает запись, написанная между апострофами в операторе вывода?*
7. *Что понимается под форматом вывода и для чего он нужен?*

Упражнения

1. Напишите следующие выражения с помощью оператора присваивания:

а) a= 48; b= 51;

б) x= 0; a= 3,6x+sinx;

в) g= 4; g= g+16;

г) a= 9,81; m= 50; F= m a;

д) x= 1; $y=\dfrac{x-63}{21-7x}$;

е) z= 25; $z=\sqrt{z}$.

2. Напишите результат работы оператора вывода на экран.

а)
```
write('a='); write(2+3);
write('='); write('2+3');
```

б)
```
writeln('a='); write(2+3);
write('='); writeln('2+3');
```

в) write('a='); writeln(2+3);
write('='); writeln('2+3');

г) write('a='); write(5);
writeln('='); write('2+3');

д) write('a='); writeln(5);
writeln('='); write('2+3');

е) writeln('a='); writeln(5);
write('='); write('2+3');

3. Определите промежуточные значения переменных и все результаты, выводимые на экран, в следующих фрагментах программ:

а) a:= −cos(pi)−sin(pi/2); x:= x*x + a;
writeln('a=',a, 'x=',x);

б) a:= 'Men '; g:= 'mustaqil ';
b:= 'O‘zbekiston'; m:= ' farzandiman!'
write(a, g, b, m);

в) a:= 9; b:=a+a;
a:= a*a−b;
write('a= ', a); write(' b=', b);

Урок 22. Повторение темы «Операторы присваивания и вывода информации на экран»

1. Напишите следующие выражения с помощью оператора присваивания:

а) $y = \frac{x-21}{7-x^{63}}$; б) a = 3,6$x$ + sin x; в) $z = \sqrt{x-5y+xtgx}$;

г) S = πr^2; д) F = ma; е) $S = \frac{ah}{2}$.

2. Определите результат работы оператора вывода.

а) a:=123.45;
write('a=', a:2:1);

б) a:=123.45;
write('a=', a:5:1);

в) a:= '2011';
writeln(a:3, ' yil':3);

г) a:= '2011';
writeln(a:4, ' yil':5);

3. Напишите вместо знака вопроса необходимые стандартные функции так, чтобы в программе переменные соответствовали описанным типам. Определите результаты, выводимые на экран.

а) var a, b, c: integer;
begin a:=25; b:= ?(sqrt(a)); c:= ?(a/b);
writeln(a, ' ', b, 'c= ', c);
End.

б) var x, y, z: word;
begin x:=?(?(−7.21)); y:= ?(sqrt(x*x));
z:= ?(x+y−100); write(z−x, y);
End.

Урок 23. Оператор ввода данных в память компьютера в диалоговом режиме

На языке Паскаль можно присваивать значения к переменным другими способами, отличающимися от оператора присваивания. Один из них называется **ввод данных в диалоговом режиме** и осуществляется с помощью **оператора ввода**.

Оператор ввода используется для присваивания значений переменным через клавиатуры в процессе работы программы. Общий вид оператора ввода:

Read (список переменных); и ReadLn(список переменных) где **Read** (англ. – читать) и **ReadLn** служебные слова языка Паскаль, список переменных — одна переменная или последовательность переменных, разделенных запятыми. Например, Read(a); Read(alfa, beta); ReadLn(_name);.

Оператор ввода приостанавливает выполнение программы и ожидает ввода значений с клавиатуры для переменных из списка. Если в списке несколько переменных, то их значения можно ввести, разъединив пробелом или нажатием клавишу ENTER. В обоих случаях после ввода всех данных нажимается клавиша ENTER.

Отличие в работе операторов Read и ReadLn заключается в следующем. Значения переменным, вводимым с помощью одного или нескольких операторов Read, можно ввести в одной строке, разъединив пробелом.

Оператор ReadLn разрешает ввести, разъединив пробелом, значения переменным только из своего списка.

Поэтому, после окончания списка переменных оператора ReadLn для работы следующего оператора ввода обязательно нажимается клавиша ENTER.

Пример 1.

```
Var a,b:Integer;
Begin
Read(a);
Read(b);
WriteLn('a+b=', a+b);
    End.
```

Пример 2.

```
Var
a,b:Integer;
Begin
Read(a, b);
WriteLn('a+b=', a+b);
    End.
```

В обоих примерах можно ввести значения одним из следующих способов.

1 способ: после запуска программы в начале строки экрана появится курсор, и программа ожидает ввода значения **a**. Например, в качестве значения **a** вводим 10, и после нажатия пробел в качестве значения **b** вводим 11. Теперь после нажатия ENTER на экране отражается следующее:

10 11
a+b=21

2 способ: после запуска программы в начале строки экрана появится курсор, и программа ожидает ввода значения **a**. Например, в качестве значения a вводим 10, и после нажатия ENTER в качестве значения **b** вводим 11. Теперь после нажатия ENTER на экране отражается следующее:

10
11
a+b=21

Пример 3.

```
Var a,b,g,m:Integer;
Begin
Read(a, b);
Read(g); m:=a+g+b;
WriteLn('Natija= ', m);
    End.
```

Пример 4.

```
Var a,b,g,m:Integer;
Begin
Readln(a, b);
Read(g); m:=a+g+b;
WriteLn('Natija= ', m);
    End.
```

В примере 3 можно ввести значения 1 или 2 способом.

В примере 4 значения переменных **a** и **b** можно ввести с помощью клавиш пробел или ENTER. Для ввода значения переменной **g** после ввода значения переменной **b** обязательно нажимается клавиша ENTER. Тогда получим один из следующих случаев.

10 11

12
Natija= 33

10
11
12
Natija= 33

Если для присвоения значений переменным использован оператор присвоения, то программа каждый раз выполняется для одного и того же значения. В этом случае для замены значения переменной приходится ввести изменение в программе. Если для присвоения значений переменным использовать оператор ввода, то при каждом запуске программы значения переменных можно ввести через клавиатуры, т.е. данные вводятся в **диалоговом режиме.**

В диалоговом режиме есть одно неудобство – приходится помнить переменную, которой присваивается значение. Чтобы избежать это неудобство, можно использовать оператор **Write** или **WriteLn**.

Например, если написать Write('a= '); ReadLn (a); Write('b= '); ReadLn(b);, то можно будет увидеть на экране, какой из переменных присваивается значение.

Оператор Readln предоставляет еще одно преимущество. Как известно, программа на языке Паскаль выполняется столь быстро, что мы не успеваем увидеть результат.

Как было упомянуто, для просмотра результатов используется пара клавиш **ALT+F5.**

Вместо того, чтобы заставить программу ждать, пока мы просматриваем результаты программы, перед оператором **end**. нужно добавить оператор **ReadLn** без списка.

В этом случае для завершения программы нажимается клавиша ENTER. Чтобы получить нужный эффект, необходимо, чтобы последний оператор ввода перед оператором без списка также имел окончание LN.

Вопросы и задания

1. *Объясните задачи оператора ввода в диалоговом режиме.*
2. *Почему этот способ называется вводом данных в диалоговом режиме?*
3. *Чем отличаются операторы ввода данных?*
4. *Объясните преимущества оператора ввода перед оператором присваивания.*
5. *Каким способом можно легко узнать переменную, для которой вводится значение?*
6. *Объясните преимущества оператора Readln для просмотра результатов программы.*

Упражнения

1. Составьте программу для вычисления квадрата *a* числа *N* для значений 0; 11; 12; 13; 14; 15 числа *N*.

2. В следующей программе ввести для переменной *a* такие значения, как «O‘ZBEKISTONIM»; «VATANIM»; «ONA DIYORIM», удобным способом и получите результат.

```
Var a, b, g: string;
Begin
        b:= ' – SAJDAGOH KABI'; g:= ' MUQADDASDIR!';
        write(a, b, g); readln;
End.
```

3. Автомобиль «Матиз» за *T* секунды проехал *S* метров. С помощью диалогового режима составить программу, которая вычисляет среднюю скорость (в м/с) для следующих значений (подсказка: $V = S/T$).

а) T = 10; S = 150;
б) T = 12; S = 200;
в) T = 20; S = 400;
г) T = 45; S = 900.

Урок 24. Повторение темы «Оператор ввода данных в память компьютера в диалоговом режиме»

1. В следующей программе ввести для переменной a, такие значения, как «ozod»; «obod», удобным способом и получите результат.

```
Var a, b, g: string;
Begin
  b:= 'Bizdan  '; g:= ' Vatan qolsin!'; write(b, a, g); readln;
End.
```

2. В следующей программе вместо вопросительного знака введите текущий год с помощью переменной m и получите результат – поздравление с годовщиной независимости.

```
Var a, b, g: string; m: word;
Begin
   write('Vvedite tekushiy god: '); ?;
a:= 'Mustaqillikning  '; g:= ' yilligi bilan '; b:='tabriklaymiz!';
writeln(a); writeln(m-1991, g); write(b); readln;
End.
```

3. Если воздействующая сила F, ускорение тела a, то составьте программу вычисления массы тела с помощью оператора ввода и получите результаты (подсказка: m = F/a).

а) F = 15, a = 55; б) F = 55, a = 15;

в) F = 10, a = 100; г) F = 100, a = 10.

4. Пусть: a=19; b=2; d=1950. Составьте программу для вычисления следующих выражений. Выберите удобный метод присвоения значений переменным.

а) $y = a + b^2 + ad$;

б) $t = \sqrt{a+b} - \sqrt[3]{d-a}$;

в) s = b cos a + sin d;

г) $n = \pi d^2 + ab$.

5. Составьте с помощью оператора ввода программу для вычисления площади треугольника со сторонами a, b, c и получите результат.

а) a = 5, b = 7, c=4;

б) a = 8, b = 6, c=10;

в) a = 3, b = 4, c=5;

г) a = 10, b = 8, c=10.

6. Составьте программу для вычисления значений функции $y = 23x+1$, при значениях $x = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$. Выберите удобный метод ввода данных.

7. Составьте программу для вычисления значений функции $y = 21x^2 + 7x + 1963$, при значениях $x = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$. Выберите удобный метод ввода данных.

Урок 25. Работа с экраном в текстовом режиме

На предыдущих уроках мы познакомились с методами вывода информации на экран. При этом каждая информация выводилась на экран в продолжение предыдущей. На языке Паскаль имеется возможность вывода информации с указанного места экрана. Кроме того, можно использовать разные цвета для текста и фона. Как говорилось выше, функции и процедуры языка Паскаль для работы с экраном, находятся в модуле **Crt**. Поэтому для применения ее функций и процедур в начале программы вводится указание **Uses Crt**;.

Модуль **Crt** предоставляет возможность вывода на экран информации в цветном формате. На языке Паскаль обычно используются 16 цветов. Каждый цвет закодирован числами от 0 до 15.

В модуле **Crt** для каждого цвета выделена константа, причем имена констант соответствуют названию цвета на английском языке.

В следующей таблице указаны цвета, их коды и имена соответствующих констант:

Цвет	Код	Имя константы
Черный	0	Black
Синий	1	Blue
Зелёный	2	Green
Бирюзовый	3	Cyan
Красный	4	Red
Малиновый	5	Magenta
Коричневый	6	Brown
Светло-серый	7	LightGray

Цвет	Код	Имя константы
Темно-серый	8	DarkGray
Голубой	9	LightBlue
Ярко-зелёный	10	LightGreen
Ярко-бирюзовый	11	LightCyan
Ярко-красный	12	LightRed
Ярко-малиновый	13	LightMagenta
Жёлтый	14	Yellow
Белый	15	White

Для выбора некоторого цвета с помощью процедуры в скобке задается код цвета или имя соответствующей константы. Целесообразно выбрать разные цвета для текста и фона, в противном случае текст не будет виден внутри фона. Последний выбранный цвет для текста и фона называется

соответственно «текущий цвет текста» и «текущий цвет фона». Если цвета не будут выбраны, то текущими цветами считаются для текста – белый, а для фона – черный.

Рассмотрим некоторые процедуры модуля **Crt**. Процедуры для работы с цветом текста и с цветом фона текста следующие:

Процедура назначения цвета текста	Процедура назначения цвета фона текста
TextColor(rang);	**TextBackGround(rang);**

здесь «rang» — переменная или постоянная, указывающая код или имя константы выбранного цвета.

Пример 1.

```
Uses Crt;
Begin
TextColor(14); {или TextColor(yellow)}
WriteLn('Данный текст имеет желтый цвет');
End.
```

Данный текст имеет желтый цвет

Пример 2.

```
Uses Crt;
Begin
 TextColor(Yellow); TextBackGround(Blue);
WriteLn(' Данный текст имеет на синем фоне желтый цвет');
End.
```

Данный текст имеет на синем фоне желтый цвет

Иногда трудно найти необходимую информацию из-за заполнения экрана данными. Примененная в этом случае процедура **ClrScr** очистит экран и установит курсор в начале (в левом верхнем углу) экрана. Если процедуру ClrScr написать после процедуры назначения цвета фона текста, то экран закрашивается в цвет текстового фона.

Пример 3.

```
Uses Crt;
Begin
   ClrScr;  {экран очищается, курсор устанавливается в начале экрана}
      TextColor(14); TextBackGround(2);
  WriteLn('Данный текст имеет на зеленом фоне желтый цвет');
End.
```

Данный текст имеет на зеленом фоне желтый цвет

Пример 4.

```
Uses Crt;
Begin
      TextColor(14); TextBackGround(2);
    ClrScr;  {экран очищается, курсор устанавливается в начале
экрана, и экран закрашивается в зеленый цвет}
  WriteLn('Данный текст имеет на зеленом экране желтый цвет');
End.
```

Данный текст имеет на зеленом экране желтый цвет

Для оформления результата программы кроме цветовых эффектов можно использовать возможность вывода результата на указанном месте. В этом случае следует установить курсор в нужном месте экрана.

На языке Паскаль такую возможность предоставляет процедура **GotoXY**. Процедура имеет следующий вид: **GotoXY(A,B)**;, где **A** и **B** целочисленные переменные или константы.

Процедура GotoXY(A,B) устанавливает курсор на пересечении A столбца и B строки экрана. В текстовом режиме экран, в основном, имеет размер 80×25, т.е. 80 столбцов и 25 строк (с помощью специальных операторов размер экрана можно изменить).

Пример 5.

```
Uses Crt;
Begin
 ClrScr;    {экран очищается}
 GotoXY(22,12);   {курсор устанавливается на пересечении 22
столбца и 12 строки}
  Write('Этот текст выводится от середины экрана');
End.
```

Этот текст выводится от середины экрана

Следовательно, в данной процедуре должны быть выполнены условия: $1 \le A \le 80$ и $1 \le B \le 25$.

В этой программе выводимый на экран текст содержит 39 символов. Чтобы вывести текст от середины экрана, нужно оставить с обеих сторон текста одинаковую позицию, значения **A** и **B** в процедуре **GotoXY** вычислены следующим образом:

A=[25/2]=12, B=[(80-39)/2]=20.

Вопросы и задания

1. *Какой модуль языка Паскаль используется для работы с экраном?*
2. *Каким количеством цветовой гаммы располагает язык Паскаль?*

3. *Какая процедура служит для выбора цвета текста?*
4. *Какая процедура служит для выбора цвета фона текста?*
5. *Для чего служит процедура ClrScr? Объясните примерами.*
6. *Сколько столбцов и строк в текстовом режиме экрана?*
7. *Возможен ли вывод результата на указанном месте экрана? Объясните ответ.*

Упражнения

1. Определите цвет экрана, цвет фона текста, цвет текста и позиции текста в результате работы следующей программы:

```
Uses crt;
Begin
    textbackground(yellow); WriteLn('O‘zbekiston'); clrscr;
    textcolor(4); write('kelajagi '); textbackground(blue);
    writeln('buyuk'); textcolor(2); write('DAVLAT!'); readln;
End.
```

2. Вывести на экран текст «O‘zbekiston konstitutsiyasi — erkinlik posboni» выбрав красный цвет для текста, синий цвет для фона текста.

3. Добавьте к следующей программе такие процедуры, чтобы на экране весь текст отражался в синем цвете, а фон текста – в желтом цвете. Объясните работу программы.

```
Var a,b: string; m, s : real;
Begin
    a:= 'Vvedite storonu kvadrata: '; в:= 'Ploshad kvadrata: ';
Write(a); readln(m); s:=sqr(m); write(b, s:8:2, ' kvadratnaya yedinitsa');
readln;
End.
```

4. Введите заданные значения переменных A и B, и объясните работу программы.

```
Uses Crt;
Var a,b: integer;
Begin  ClrScr;  write('A= '); ReadLn(a); write('B= '); ReadLn(b);
GotoXY(A,B);  WriteLn('Kitob bilim manbai'); ReadLn;
End.
```

а) A=1, B=1; б) A=8, B=1; в) A=1, B=8;
г) A=8, B=8; д) A=25, B=25; е) A=100, B=10.

5. Выведите на экран свою фамилию, имя и отчество 3 разными цветами на 3 разных цветах фона и в разных позициях.

Урок 26. Повторение темы «Работа с экраном в текстовом режиме»

1. Вывести на экран текст «Ajdodlar merosini qadrlaylik», выбрав синий цвет для текста, зеленый цвет для фона текста.

2. Вывести на экран текст «Vatanni sevmoq iymondandir!» с правой стороны экрана на 12 строке зеленым цветом на красном фоне.

3. Составьте программу, которая обеспечивает вывод текста на экран по смыслу текста «Самая верхняя строка, слева», «Самая верхняя строка, справа», «Самая верхняя строка, посередине», «Самая нижняя строка, слева», «Самая нижняя строка, справа», «Самая нижняя строка, посередине», «Строка в центре, слева», «Строка в центре, справа», «Строка в центре, посередине».

4. Выведите на экран фразу «Suv — hayot manbayi» 5 разными цветами в разных позициях.

5. Выведите на экран фамилии пяти одноклассников разным цветом на желтом фоне.

Урок 27. Составление линейных программ

Обычно, запись линейного алгоритма на языке программирования называется **линейной программой**.

То есть, в линейных программах операторы выполняются последовательно один за другим, не меняя порядок, и никакое условие не проверяется.

Пример 1. Составить программу вычисления длины окружности радиуса R и получить результат при R = 9 единиц.

Решение. Вспомним формулу вычисления длины окружности: L=2 π R. На языке Паскаль она пишется в виде L:=2*pi*R. В программе участвуют одна константа **pi** и две переменных **R** и **L**. Согласно условиям задачи R = 9, т.е. целое число.

Следовательно, определим тип переменного R как Integer. Длина окружности L относится к вещественному типу Real, так как в произведении участвует число π. С учетом изложенных соображений, составим следующую программу:

```
Program dlina_okrujnosti;
Var  R:Integer;  L:Real;
Begin
   r := 9; L := 2*pi*R; WriteLn('L=',L,' единиц'); readln;
End.
```

После запуска программы (после нажатия клавиш **Ctrl+F9** или выбора команду **Run** в меню **Run**) на экране появится следующий результат: **L= 5.6548667765E+01 единиц**.

Поскольку L типа real, результат получится в экспоненциальной форме. Если применить формат L:7:2 в операторе вывода, то увидим, что длина окружности радиусом 9 единиц равна 56,54 единицам.

С помощью этой программы можно вычислить длину любой окружности с целочисленным радиусом. При этом каждый раз в прог-

рамме приходится изменять значение радиуса **R**. Чтобы каждый раз не изменять программу, удобнее вводить значение радиуса **R** в диалоговом режиме. Учитывая, что значение радиуса не всегда может быть целым числом, опишем его как переменная типа **Real**. На основе этих рассуждений составим следующую программу:

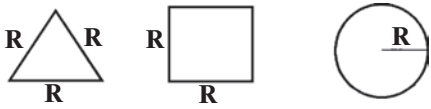
Программа	**Результат на экране**
Program dlina_okrujnosti; Var r, L : Real; Begin Write('Введите радиус '); ReadLn(r); L := 2*pi*r; WriteLn('L= ',L,'b единиц); readln; End.	Введите радиус: 9 L=5.6548667765E+01 единиц

После запуска программы на экране появится запись «Введите радиус:» и курсор останется на этой строке. Оператор ReadLn остановит работу программы пока не будет введено значение переменной **R**. После ввода через клавиатуры числового значения 9 радиуса и нажатия клавиши ENTER программа принимает 9 как значение переменной **R** и продолжает работу. Оператор WriteLn выводит искомый результат, и последний оператор ReadLn остановит работу программы пока не будет нажата клавиша Enter. Запуская программу заново и вводя различные значения радиуса, получим длины окружностей с различными радиусами.

Пример 2. Составьте программу, которая вычисляет с помощью формулы Герона площадь треугольника со сторонами a, b, c и получите результаты при $a = 3$, $b = 4$, $c = 5$.

I способ	**II способ**
Program Ploshad_treugolnika; Var a,b,c:Integer; {стороны треугольника} yp,s:Real; {yp−половина периметра, s-площадь} Begin a:=3; b:=4; c:=5; yp:=(a+b+c)/2; s:=sqrt(yp*(yp−a)*(yp−b)*(yp−c)); WriteLn('S= ',s,' кв.единиц'); ReadLn; End.	Program Ploshad_treugolnika; Var a,b,c:Integer; {стороны треугольника} yp,s:Real; {yp−половина периметра, s-площадь} Begin Write('Введите a,b,c'); ReadLn(a,b,c); yp:=(a+b+c)/2; s:=sqrt(yp*(yp−a)*(yp−b)*(yp−c)); WriteLn('S=',s:2:2,' кв.единиц'); ReadLn; End.
S = 6.0000000000E+00 кв.единиц	Введите a, b, c 3 4 5 S = 6.00 кв.единиц

Пример 3. Составьте программу, которая вычисляет площади равностороннего треугольника, квадрата со сторонами *R* и круга с радиусом R. Получите результат при R = 4.



Программа	Результат на экране

Program Ploshadi;
var r: Integer; s1,s2,s3:Real;
begin
Write('Введите R :');ReadLn(r);
s1:=sqr(r)*sqrt(3)/4; s2:=sqr(r);
s3:=pi*sqr(r);
WriteLn('Площадь треугольника = ',s1);
WriteLn('Площадь квадрата = ',s2);
WriteLn('Площадь круга = ',s3); ReadLn;
end.
	Введите R : 4
	Площадь треугольника =
	6.9282032303E+00
	Площадь квадрата =
	1.6000000000E+01
	Площадь круга =
	5.0265482457E+01

Составьте программу этой задачи с помощью оператора присваивания в качестве самостоятельной работы.

Вопросы и задания

1. *Что понимается под линейной программой?*
2. *Пользуясь какими операторами составляют диалоговые программы?*
3. *Почему не обязателен ввод значения к числу π?*
4. *Расскажите о формате вывода.*
5. *Какие клавиши нужно нажать, чтобы увидеть результат на экране?*

Упражнения

1. Объясните работу следующих линейных программ и определите результат.

а)
```
Var a,b:String;
Begin
a:= 'O`zbekiston ';
b:= 'Davlat';
WriteLn(a, 'Mustaqil ', b);
End.
```

б)
```
var a,b:Integer;  s:Real;
Begin a:=4;
a:=sqr(a); b:=b−a;
s:=2*a+3*b;
WriteLn('S= ',s);
End.
```

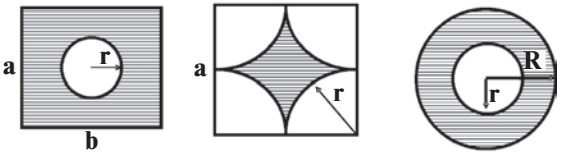
2. Внизу приведена программа вычисления площади трапеции в не упорядоченном виде. Восстановите программу.

p := (a+b) /2; s := p*h; Program Ploshad_trapetsii; End. WriteLn('S ='; s,' kv.yedinits ');
ReadLn(a,b,h); Begin Write('Vvedite A,B,H : '); Var a,b,h: Integer;
p,s: Real;

3. Даны стороны треугольника a, b и угол α между ними. Составьте программу для вычисления площади треугольника (подсказка: $s = \frac{1}{2}\, a \cdot b \cdot \sin \alpha$).

Урок 28. Повторение темы «Составление линейных программ»

1. Составьте программу, которая вычислит площади заштрихованной части нижеследующих геометрических фигур (подсказка: площадь какой фигуры следует отнимать от площади другой фигуры?).



2. Составить программу, которая поменяет местами значения переменных **a** и **b** целого типа, т.е. при a = 7 и b = 2 должны получить a = 2 и b = 7 (подсказка: m = a, a = b, b = m).

3. Составить программу, которая вычисляет сопротивление **R** при параллельном соединении проводников с сопротивлениями $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, $\mathbf{R_4}$. (подсказка: $1/R=1/R_1+1/R_2+1/R_3+1/R_4$).

Урок 29. Операторы перехода и ветвления

До сих пор мы рассматривали линейные программы, указания которых выполняются строго одно за другим. Иногда для решения поставленной задачи приходится нарушать порядок выполнения операторов, т.е. в программе управление передается назад или вперед. В связи с этим перед оператором, которому передается управление, ставится **метка**. Метка, как имя переменной, состоит из латинских букв и цифр. Например, 7, N1, metka2.

Для имени метки можно использовать числа от 0 по 9999. Опережающие нули не считаются.

Используемые в программе метки должны быть перечислены в части описаний программы с помощью служебного слова **Label**. Метки используются только в случае применения **оператора перехода**. Оператор перехода имеет вид: **GOTO < метка >**;, GOTO (англ. – перейти к). Оператор перехода передает управление оператору с соответствующей меткой.

Пример 1: Label N1; Var a,b,c:Integer; Begin a:=15; b:=13; c:=a+b; GoTo N1; {управление передано оператору с меткой N1} c:=a−b; N1: WriteLn(c); End.	В результате работы программы значение переменной с равно 28. Это случилось из-за того, что управление передано оператору вывода с меткой N1 и пропущен оператор c:= a−b.

Оператор перехода передает управление оператору с меткой не зависимо от выполнения условий. Но в большинстве случаев приходится выбирать ту или иную последовательность действий в зависимости от выполнения некоторых условий. Например, при решении квадратного уравнения в зависимости от знака дискриминанта приходится выбрать одно из трех направлений. Наверно, вы уже вспомнили **разветвляющиеся алгоритмы.** Для решения подобных задач на языке Паскаль предусмотрен **оператор ветвления**.

Оператор ветвления имеет следующий общий вид:

If <условие> **Then** <оператор или операторы> **Else** <оператор или операторы>;

Здесь **If , Then, Else** − служебные слова языка Паскаль, которые читаются и означают следующее: **If** (иф) — «если», **Then** (зен) — «то», **Else** (элз) — «иначе». Обычно, <условие> — логическое выражение, которое принимает одно из двух значений «истина» или «ложь»; <оператор или операторы> — один оператор или последовательность операторов. Следует запомнить, что после оператора, записанного перед служебным словом **Else**, не ставится **«;»** (точка с запятой).

Оператор ветвления работает следующим образом: сначала проверяется **условие**, если значение условия «истина», то выполняются оператор или операторы, следующие за служебным словом **Then**, иначе выполняются операторы, следующие за служебным словом **Else**.

Пример 2. Составить программу, которая выводит фразу «Больше», если введенное число больше 25, в противном случае фразу «Не больше».

Решение: Так как не указан тип числа, опишем его как вещественный.

```
Program Sravneniye;
Var a:Real;
Begin Write('Введите любое число: '); ReadLn(a);
  If  a>25 Then WriteLn('Больше')
                Else WriteLn('Не больше');
End.
```

Части оператора ветвления можно писать в отдельных строках.

Если после THEN или ELSE записаны несколько операторов, то между ними ставится символ «;», а сами они заключаются между служебными словами **begin** и **end**. Такая конструкция называется **блок операторов.**

Пример 3. Составить программу, которая вычисляет произведение и отношение заданного числа *a* на число *b*.

Решение: Так как не указан тип чисел, опишем их как вещественные.

```
Program Otnosheniye;
Label konets;
Var a, b: Real;
Begin
   Write('Введите число a: '); ReadLn(a); Write('Введите число b: ');
ReadLn(b);
    WriteLn('Произведение: ', a*b);
    If  b=0 Then begin WriteLn('Нельзя делить'); goto konets; end;
    WriteLn('Отношение: ', a/b);
    konets:
End.
```

Часть **Else** оператора ветвления используется по обстоятельствам, т.е. оператор ветвления можно использовать в следующем виде:

If <условие> **Then** <оператор или операторы>

Эта запись называется **короткой** формой оператора ветвления. В этом случае, если значение условия «истина», то выполняются оператор или операторы, следующие за служебным словом **Then**, иначе управление передается следующему (после оператора ветвления) по записи оператору.

Пример 4. Составить программу, которая заменит число его кубической степенью, если заданное целое число отрицательное.

Решение:

```
var a:Integer;  {заданное число}
begin   Write('Введите любое целое число: '); ReadLn(a);
  If a<0 Then a:= a*a*a; {если число отрицательное, то заменяется
кубической степенью}
WriteLn(a); readln; {оператор readln добавлен для просмотра результата}
End.
```

В составе оператора ветвления можно использовать оператор ветвления.

Пример 5. Составьте программу для определения знака числа.

```
Var a:Integer; b:String; {в одной строке можно описать несколько
переменных}
Begin
  Write('Введите любое число: '); ReadLn(a);
```

```
If a<0 Then b:= 'отрицательное ' Else If a>0 Then b:= 'положительное' Else b:= 'нуль';
  WriteLn(b);
End.
```

Пример 6. Составьте программу, которая определит наибольшее из чисел a и b.

```
Var a,b,maks:Real;
Begin
  Write('Введите первое число = '); ReadLn(a);
  Write('Введите второе число = '); ReadLn(b);
  If a>b Then maks:=a Else maks:=b; WriteLn('Наибольшее из чисел = ',maks);
End.
```

Вне зависимости от выполнения условия a>b оператор WriteLn('Наибольшее из чисел = ',maks); обязательно выполнится. Потому что он не входит в состав оператора ветвления, хотя в программе он написан на одной строчке с ним. Найдите объяснение!

Вопросы и задания

1. *Для чего служит метка?*
2. *Какой общий вид оператора перехода?*
3. *Может ли в программе участвовать оператор перехода, но отсутствовать метка?*
4. *Для чего служит оператор ветвления?*
5. *Какими служебными словами начинается и заканчивается блок операторов?*
6. *Расскажите о полном и сокращенном виде оператора ветвления?*
7. *После какого оператора не ставится знак «;»?*

Упражнения

1. Определите ошибочные записи среди следующих операторов перехода:

а) Goto 10; б) goto 30; в) goto −5;
г) GoTo _5; д) goto sin; е) goto 2_5; ж) GOTO a_5;

2. Определите ошибочные условия для оператора ветвления.

а) a<>b; б) a<−b; в) a><b;
г) −a>0; д) −1>0; е) a>>b; ж) a:=b;

3. Определите ошибочные записи.

а) IF a=b THEN a:=a+1; ELSE b:=a;
б) IF a:=1 THEN a:=a+1 ELSE b:=a;

4. Составьте программу, которая вычисляет значение следующей функции для заданных значений *x*: $y = \begin{cases} -1, если\ x > 0, \\ x^2, если\ x \le 0. \end{cases}$

5. Даны три числа. Составьте программу, которая вычисляет кубическую степень отрицательных среди этих чисел.

6. Составить программу, которая вычисляет модуль заданного целого числа, если оно отрицательное. Программу составить двумя способами: с применением функции abs(x) и без нее.

7. Составьте программу для проверки правильности вводимого пароля «informatika».

Урок 30. Повторение темы «Операторы перехода и ветвления»

1. Для заданных значений определите значение условия.
   а) a:=10; b:=a*3; условие: «a<b/3»;
   б) a:=10; b:=a*3; условие: «a<=b/3»;
   в) a:=10; b:=a; условие: «a+b=2*b»;
   г) a:=10; b:=a+3; условие: «a+3>=b−3»;
2. Определите значения переменных на основе разветвления.
   а) aa:=7; bb:=6.6; if aa=round(bb) then mm:='Ha' else mm:='Yo‘q';
   б) ag:=true; if ag then aa:=21 else aa:=7; a:=a+1963;
   в) ag:=true; if ag then aa:=21 else begin aa:=7; a:=a+1963; end;
   г) ms:=50; aa:=10; if ms div aa = aa*5 then ms:=trunc(ms/3) else aa:= ms mod aa;
3. Даны числа *a* и *b*. Составить программу, которая заменит число *b* нулем, если число *b* меньше числа *a*, в противном случае оставит *b* без изменения.
4. Даны три числа *a*, *b* и *c*. Составить программу, которая вычислит квадратный корень только положительных чисел.
5. Составить программу, которая вычислит корень уравнения для следующих значений *a*, *b*.
   а) a= −1, b=1; б) a=0, b=4;
   в) a=1, b=0; г) a=1, b= −5.
6. Дано целое число *A* и натуральное число *B*. Составить программу, которая определяет, делиться ли число *A* на число *B* без остатка.
7. Даны три числа *a*, *b* и c. Составьте программу, которая вычисляет сумму этих чисел, если выполняется условие $a^2+b^2=c^2$, в противном случае — произведение модулей.

Урок 31. Составление программ со структурой ветвления

На предыдущем уроке мы рассмотрели составления программ на основе простых условий. В операторе ветвления можно использовать и составные условия. Составные усло-

вия получаются из простых условий при помощи логических операций **NOT** — «нет», **AND** — «и», **OR** — «или».

Вы знаете, что **NOT** — логическое отрицание, **AND** — логическое умножение и **OR** — логическое сложение. Эти операции вам знакомы из 8 класса. Операция **NOT** отрицает значение условия, записанного после себя. Операция **AND** выдает истинное значение, если условие, на которое действует эта операция, истинно.

В логических выражениях в первую очередь выполняется операция **NOT**, во вторую очередь операция **AND** и в третью очередь операция **OR**. Если в логических выражениях участвуют скобки, то в первую очередь выполняются операции внутри скобок. Последовательно идущие равноправные операции выполняются слева направо. На языке Паскаль простые условия, входящие в состав составных условий, заключаются в скобки.

Например,

1) x ∈ [a,b] (т.е., $a \le x \le b$) на языке Паскаль: (A<=X) **AND** (X<=B);

2) $\overline{t_1 = t_2}$ на языке Паскаль: **NOT**(T1=T2);

3) y<−5 или y>2 на языке Паскаль: (Y<−5) **OR** (Y>2).

Пример 1. Составьте программу, которая вычислит значение следующей функции для заданных значений *x*:

$$y = \begin{cases} x^2, & \text{если } x \in (0,1], \\ x, & \text{если } x \notin (0,1]. \end{cases}$$

Решение:

```
Var x, y: real;
Begin   Write('x= '); ReadLn(x);
          If (0<x) And (x<=1) Then  y:=Sqr(x)  Else y:=x;
         WriteLn('y= ',y): ReadLn;
End.
```

Здесь при выполнении обоих условий 0<*x* и *x*<=1, т.е. для значений x из промежутка (0,1], выполняется оператор присвоения y:=Sqr(x), в противном случае, т.е. для значений x из промежутка (0,1] выполняется оператор присвоения *y*:=*x*.

Следующие примеры раскрывают суть составных условий:

1) If (*A*>*B*) And (*B*>*C*) Then S:=B+7 ELSE S:=A*B−1;

Если *A*>*B* и *B*>*C*, т.е. *A*>*B*>*C*, то выполняется оператор S:=B+7, в противном случае S:=A*B−1.

2) If 5*B=M*M Then Goto 200 ELSE Goto 400;

Если 5*B=M*M, то управление передается оператору с меткой «200», в противном случае – оператору с меткой «400».

3) If R1<=R2 Then begin WriteLn(S); R:=R1+R2 end
Else begin WriteLn(S*R1); R1:=R2; R2:=0; end;

Если R1<=R2, то выполняются операторы WriteLn(S) и R:=R1+R2, в противном случае операторы WriteLn(S*R1), R1:=R2 и R2:=0.

4) If SS ='ZELYONIY' Then WriteLn('Можно проходить') Else WriteLn('Нельзя проходить ');

Если значение SS (цвет светофора) «ZELYONIY» (зеленый) выполняется оператор WriteLn('Можно проходить'), в противном случае оператор WriteLn('Нельзя проходить').

Теперь рассмотрим программы со структурой ветвления:

Пример 2. Составьте программу, которая определяет наибольшее из чисел *a*, *b* и *c*.	**Пример 3.** Составьте программу, которая определяет четное или нечетное натуральное число.
Program UKT; Var a,b,c,max : Real; Begin Write('Введите значения чисел a,b,c: '); ReadLn(a,b,c); If a>b Then max:=a Else max:=b; If c>max Then max:=c; WriteLn('Наибольшее из трех чисел= ',max); End.	Program Toq_juft; Var n : word; Begin Write('Наибольшее из чисел :'); ReadLn(n); If Odd(n) Then WriteLn('Нечетный') Else WriteLn('Четный'); End.

Пример 4. Составьте программу, решающую квадратное уравнение $ax^2 + bx + c = 0$.

```
Program Kv_urav;
Label Konets;
Var a,b,c,d,x1,x2 : Real;
Begin
  Write('Введите значения a,b,c: '); ReadLn(a,b,c);
  d:=Sqr(b) – 4*a*c;  {вычислен дискриминант}
   If d<0 Then begin WriteLn('Нет решения'); Goto Konets; end;
     If d=0 Then begin WriteLn('Решение единственное: ');
WriteLn('x= ', –b/(2*a));
   Goto Konets; end;
```

```
    WriteLn('Два решения : ');
    x1:=(−b−Sqrt(d))/(2*a);  x2:=(−b+Sqrt(d))/(2*a);
    WriteLn('x1= ',x1); WriteLn('x2= ',x2);
Konets: ReadLn;
End.
```

Решения рассмотренных задач показывают, что в программах со структурой ветвления организация ветвления зависит от поставленной задачи.

Вопросы и задания

1. *Чем отличается полная форма оператора ветвления от сокращенной формы?*
2. *Какие логические операции применяются на языке Паскаль?*
3. *Приведите примеры составных условий.*
4. *Расскажите о порядке выполнения операций в логическом выражении.*
5. *В каких случаях логического выражения используются скобки?*

Упражнения

1. Найдите ошибки в следующих операторах и устраните их:
   а) IF d>0 THEN 63 ELSE s:=d+a;
   б) IF s1<>s2 THEN ELSE g1:=s1*s2;
   в) IF i*j THEN goto vo ELSE goto ne;
   г) IF x<>0 AND x<=5 THEN y=4*sin(x);
2. Определите порядок действий в следующих логических выражениях:
   а) a<−6 OR a>=0 AND a<4;
   б) x*x +y >0 AND a=0.1 OR (b>3.7 AND s<>k4);
   в) v= 'ha' AND x1>0 AND x2>0 ;
   г) a>0 OR a<1 OR NOT x*x+x*x<=1;
   д) NOT v<=b AND (f<=f1 OR t='.');
   е) NOT(NOT(NOT(a>b) OR TRUE) AND FALSE);
3. Даны длины трех отрезков. Составьте программу, которая определяет, можно ли из этих отрезков построить треугольник или нет.
4. Составьте программу, которая проверяет знания по таблице умножения учащихся младших классов. При правильном ответе вывести «Молодец», в противном случае «Работай дальше», в различных цветах текста.
5. Составьте программу, которая выводит день недели для заданной цифры из промежутка от 1 по 7.

Урок 32. Повторение темы «Составление программ со структурой ветвления»

1. Определите результат сложных логических операций.

а) a:=true; b:=true; m:=false; bb:=NOT(a AND m) AND (a OR b) OR m;

б) a:=77; b:=11; m:=7; ms:= (a div b=m) AND (a mod m=0) AND NOT((a>b) OR (b<m));

2. Определите значения переменных на основе разветвления.

а) x:= −1; y:=0; a:= 0.1; IF (x∗x +y >0) AND (a=1/10) THEN mm:=true else mm:= false;

б) x1:=sqr(-1); v:= 'h a'; x2:=sqrt(x1); IF (v= 'ha') AND (x1>0) AND (x2>0) THEN x1:=0;

3. Даны три числа *a*, *b* и *c*. Составьте программу, которая проверяет выполнение условия: $a<b<c$.

4. Даны целые числа *A* и *B*. Составить программу, которая выводит квадрат суммы чисел, если число *A* делится на число *B* без остатка, в противном случае − их произведение.

5. Составить программу, которая выводит квадратный корень заданного числа *N*, если число положительное и кратное 5, в противном случае − квадрат числа.

6. Даны числа *M* и *N*. Составить программу следующего содержания: если числа положительные и их сумма больше 100, вывести отношение числа *M* к числу *N*. Если числа положительные и их сумма не больше 100, то произведение чисел *M* и *N*.

7. Составить программу, которая выводит целую часть, если целая часть больше умноженной на 1000 дробной части, в противном случае − первые 3 цифры дробной части.

Урок 33. Оператор цикла с параметром

При решении целого ряда задач часто приходится повторять одно и то же действие. С такими задачами вы ознакомились в предыдущей главе, и для их решения составляли повторяющиеся алгоритмы. Теперь рассмотрим методы составления программ повторяющихся структур.

Для составления программ повторяющихся структур используются операторы цикла (повторения). На языке Паскаль таких операторов три и на этом уроке рассмотрим один из них — **оператор цикла с параметром.**

Общий вид этого оператора:

For I: =N1 To N2 Do <тело цикла >,

где **For** (для), **To** (по) и **Do** (выполнить) — служебные слова языка Паскаль; **I** — произвольная переменная целого типа, именуемая **параметром цикла**; **N1** — начальное значение параметра цикла; **N2** — конечное значение параметра цикла; **<тело цикла>** — оператор или последовательность операторов, которые выполняются повторно. Если тело цикла состоит из последовательности операторов, то они должны содержаться внутри begin и end;. Начальные и конечные значения параметра цикла могут быть константой, переменной или выражением.

Этот оператор работает следующим образом:

1. Сначала параметр цикла принимает начальное значение.

2. Если значение параметра цикла не больше конечного значения параметра цикла, то выполняются операторы, составляющие тело цикла, в противном случае повторение прерывается, и управление переходит к очередному оператору.

3. Значение параметра цикла увеличивается на единицу (к нему прибавляется 1 и осуществляется переход к пункту 2.

Оператор **For** применяется, главным образом, в тех случаях, когда количество повторений заранее известно.

Пример 1. Составить программу, которая выводит текст «Узбекистан — отчизна моя!» на экран 20 раз.

Решение. Согласно условиям задачи текст «Узбекистан — отчизна моя!» следует 20 раз вывести на экран, т.е. оператор «WriteLn(' Узбекистан — отчизна моя!')» должен повториться 20 раз.

Составим следующую программу:

```
Program Sikl;
Var I : Integer;
Begin
  For I:=1 To 20 Do  WriteLn('Узбекистан — отчизна моя!');
End.
```

В программе начальное значение параметра цикла I равно 1, а конечного значение 20. Тело цикла состоит из одного оператора: WriteLn('Узбекистан − отчизна моя!');. После запуска программы параметр цикла поочередно принимает значения 1, 2, 3,..., 20, и каждый раз выполняется оператор WriteLn('Узбекистан − отчизна моя!');. В результате нужный текст выводится на экран 20 раз.

Если в программе заменить начальное значение параметра цикла на 41, а конечное значение на 60, результат программы не изменится, потому что 60−41+1=20.

Пример 2. Составьте программу, которая выводит на экран числа с 1 по 20 в порядке возрастания.

Решение. Обозначаются числа переменной *S*. Сначала принимается *S*:=0. С помощью «*S*:=*S*+1» значение *S* увеличивается на единицу и выводится на экран оператором WriteLn(S). Эти операции должны повторяться 20 раз, и поэтому применятся оператор For.

```
Program Chisla;
Var I, S : Integer;
Begin
        S:=0;
      For I:=1 To 20 Do begin S:=S+1; WriteLn(S); end;
End.
```

Параметр цикла можно использовать в теле цикла. Но не рекомендуется изменять его значение в теле цикла. Легко заметить, что переменные *S* и *I* принимают одинаковые значения. Следовательно, вместо *S* на экран можно вывести *I*.

Учитывая все это, программа пишется в следующем виде:

```
Program Posledovatelnost;
Var I : Integer;
Begin
   For I:=1 To 20 Do WriteLn(I);
End.
```

Пример 3. Составьте программу, которая выводит целые числа от 1 до 100 в порядке убывания.

Решение: Выводимые на экран числа обозначим через **son**, а параметр цикла через *i*.

```
Program Chisla;
Var i,son : Integer;
Begin
  son:=101;
  For i:=1 to 100 Do Begin  son:=son−1; WriteLn(son);  end;
End.
```

В операторе **For** параметр цикла может изменяться в порядке убывания, т.е. начальное значение параметра цикла может быть больше, чем его конечное значение.

В этом случае вместо служебного слова **To** используется **Downto** (вниз к). Учитывая это, программу можно упростить:

```
Program Chisla;
      Var i : Integer;
Begin
      For i:=100 Downto 1 Do WriteLn(i);
End.
```

Пример 4. Составьте программу, которая вычислит сумму нечетных чисел от 1 по 21.

Решение. Для просмотра чисел от 1 по 21 можно использовать параметр цикла. В сумме S=1+2+3+...+21 слагаемые не превосходят 255, а значит, в этом случае для параметра определим тип **byte**. Так как значение S целое и неотрицательное, для него выберем тип **word**. Для проверки нечетности чисел удобно применить функцию Odd(x).

```
Program Summ;
Var s: word; i: byte;
Begin
      S:=0;
  For i:=1 to 21 Do If odd(i) then S:=S+i;
     WriteLn('S= ', S);
End.
```

Пример 5. Составьте программу, которая вычислит сумму элементов с четными индексами заданного массива A[1..21].

Решение. Для ввода массива и просмотра элементов используем параметр цикла, а для проверки четности индексов используем логическое выражение NOT(Odd(x)).

```
Program Massiv;
Var i: Integer; s: real;
a: array[1..21] of real;
Begin
      For i:=1 to 21 Do Begin  WriteLn('a[', i, ']= '); ReadLn(a[i]); end;
       S:=0;
  For i:=1 to 21 Do If NOT(odd(i)) then S:=S+a[i];
     WriteLn('S= ', S); ReadLn;
End.
```

Как видите, для ввода и вывода массива, просмотра индексов удобно использовать оператор цикла с параметром.

Вопросы и задания

1. *Приведите примеры повторяющихся алгоритмов.*
2. *Как выглядит оператор цикла с параметром?*
3. *Какие значения принимает параметр цикла?*
4. *Расскажите о начальном и конечном значениях параметра цикла.*
5. *Ограничено ли значение параметра цикла?*
6. *Объясните работу оператора цикла с параметром.*
7. *В каких случаях в операторе цикла вместо служебного слова To применяется Downto?*

Упражнения

1. Определите количество повторений для следующих операторов.
   а) for i:=1 to 88 do b:=1; б) for i:=73 to 161 do m:=2;
   в) for i:= −21 to 0 do a:=3;
   г) a:=5; b:=34; for i:=a+7 to b −1 do s:=s+1;

д) a:=5; b:=19; for i:=a*a to 2*b+8 do s:=s+1;

2. Составьте программу, которая вычислит значения функции $y = 21x^2 + 7x + 1963$ при $x = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$.

3. Составьте программу вычисления значений функции: $y = 23x + 1$ для целых значений x из интервала $[-15, 5]$.

4. Задан массив A[1..17]. Составьте программу, которая выводит индексы тех элементов, значения которых равны нулю.

5. Составьте программу вычисления значений функции $y = 2x + 19$ для значений x из интервала $[0, 10]$ с шагом 0.25 (подсказка: x=0 при i=0; x=0,25 при i=1; ...; i=40 при x=10).

Урок 34. Повторение темы «Оператор цикла с параметром»

1. Найдите и объясните ошибки в следующих операторах:
а) for I= −15 to 5 do s:=s+I;
б) for kub:=100/10+11 to 1963 do begin a:=7; end;
в) for mag:=99 dawnto 1 do readl(aa);
г) for bma:= 0.5 to 10 do writeln(k);

2. Определите количество повторений для следующих операторов:
а) for k:=trunc(23/5) downto trunc(1/2) do m:=1991;
б) for s:=23 to 1 do m:=1963;
в) for J:=2 downto 19 do m:=1950;
г) for d:=23 downto 1 do m:=2009;
д) for i:=abs(−25) to 25 do s:=s+i*i;
е) for h:=round(9.6) downto trunc(3*3) do a:=21;

3. Составьте программу вычисления суммы S=10+12+14+...+50.

4. Составьте программу вычисления суммы

$$S = \frac{7}{11} + \frac{17}{21} + \frac{27}{31} + ... + \frac{2007}{2011}$$

(подсказка: остаток деления J на 10 равен к 7).

5. Составьте программу вычисления произведения P=1*3*5*...*23.

6*. Задан массив A[1..5]. Составьте программу, которая выводит на экран элементы массива в обратном порядке.

7*. Задан массив A[1..15]. Составьте программу, которая выводит разность сумм элементов массива четными индексами и элементов массива с нечетными индексами.

Урок 35. Операторы цикла по условию

В рассмотренных примерах предыдущего занятия количество повторений цикла было заранее известно. Однако встречаются и такие задачи, когда повторение некоторой

последовательности действий продолжается до выполнения определенного условия. Естественно, в таких случаях заранее указать количество повторений не представляется возможным. В таких случаях используется **оператор цикла по условию**. На языке Паскаль таких операторов два – **While** и **Repeat**. Оператор **While** имеет следующий общий вид:

While <условие> Do <тело цикла>;

где **While** (англ. пока) и **Do** — служебные слова языка Паскаль; **<условие>** — простое или составное логическое выражение. Если тело цикла состоит из последовательности операторов, то они должны содержаться внутри **begin** и **end**;.

Этот оператор работает по следующей схеме:

– сначала проверяется **условие**. Если его значение **истина**, то выполняются операторы, составляющие тело цикла, затем снова проверяется **условие** и т.д.

Этот процесс продолжается до тех пор, пока условие не примет значение **ложь**.

Пример 1. Составить программу, которая вычислит наибольший общий делитель (НОД) двух целых чисел.

Решение. Самый удобный способ определения НОД — алгоритм Эвклида. Этот алгоритм вам известен с первой главы. Пользуясь оператором **While**, составляем программу:

```
Program NOD;
Var a, b : Integer;
Begin
  Write('Введите первое число: '); ReadLn(a);
  Write('Введите второе число: '); ReadLn(b);
  While a<>b Do  If a>b Then a:=a–b Else b:=b–a; {при a=b цикл завершается}
   WriteLn('NOD = ', a);
End.
```

Оператор **Repeat** тоже выполняет цикл по условию. Он имеет следующий общий вид:

Repeat
<тело цикла >
Until <условие>

где **Repeat** (англ. повторение) и **Until** (англ. до) служебные слова языка Паскаль, Repeat – обозначает начало цикла, Until – конец цикла; <условие> – простое или составное логическое выражение. Цикл повторяется пока условие не примет значение «истина».

Пример 2. Составить программу для вычисления суммы $S = 1{,}1+1{,}5+ +1{,}9+2{,}3+...+45{,}5$.

Решение. Как видно, слагаемые отличаются на 0,4. В операторе цикла с параметром параметр цикла не может быть вещественным. В таких случаях удобно использовать оператор цикла по условию.

Сравните предлагаемые вам два решения.

С помощью While	**С помощью Repeat**
Program Summ_real; Var J, S: real; Begin S:=0; J:=1.1; While J<= 45.5 do begin S:=S+J; J:=J+0.4; end; WriteLn('S= ', S); End.	Program Summ_real; Var J, S: real; Begin S:=0; J:=1.1; Repeat S:=S+J; J:=J+0.4; Until J>=45.5; WriteLn('S= ', S); End.

Пример 3. Составьте программу, которая выводит на экран букву «A» в разных цветах, используя генератор случайных чисел (Random). Программа завершит работу, когда появится буква «A» красного цвета.

Решение. Известно, что на языке Паскаль цвет текста закодирован целыми числами от 0 до 15. Функция Random(x) генерирует случайные числа из интервала [0, x). Поэтому для получения случайных чисел из интервала [0, 15], используем функцию Random(16).

Следует заметить, что функция **Random(x)** при каждом запуске программы генерирует одинаковую последовательность чисел. Чтобы избежать этого, используется оператор **Randomize**, который обычно записывается раньше функции Random. Учитывая, что код красного цвета равен 4, составим следующую программу:

```
Program Bukva;
Uses Crt;
Var rang : Integer;
Begin
  Randomize;
   Repeat
     rang:= Random(15); TextColor(rang); Write('A');
   Until rang=4;   {если rang=4 (красный), то цикл завершается}
End.
```

Первое отличие оператора Repeat от оператора While заключается в том, что в операторе While условие проверяется в начале, а в операторе Repeat в конце цикла. Поэтому, тело цикла в операторе While может вообще не выполняться (если условие в начале не выполнено), а в операторе Repeat тело цикла выполняется хотя бы один раз. Первое отличие заключается в том, что в операторе While цикл завершается в том случае, если условие не выполнено (когда значение «ложь»), а в операторе Repeat — когда условие выполнено (когда значение «истина»).

В примере 1 необходимо сначала проверить условие. Поэтому мы использовали оператор While, а в примере 3 сначала нужно определить цвет, а потом проверить условие. Следовательно, целесообразно использовать оператор Repeat. Внеся небольшие изменения в программу можно вместо While использовать Repeat, и наоборот. Правильное применение этих операторов дает больше эффекта.

Вопросы и задания

1. *Какие операторы цикла по условию вам известны?*
2. *Расскажите о работе оператора While.*
3. *Расскажите о работе оператора Repeat.*
4. *Чем отличаются операторы цикла по условию и оператор цикла с параметром?*
5. *В каких случаях эффективнее использовать тот или иной оператор цикла?*

Упражнения

1. Найдите и объясните ошибки в следующих операторах:
   а) whIle 5*6 do SH:=sqr(2);
   б) WHILE 5>6 do Od:=Od+1;
   в) Repeat i<j Until s:=0;
   г) rEpEaT s:=0 UntiL s:=0;
2. Определите количество повторений для следующих операторов:
   а) x:= −5; while X>0 do x:=x+2;
   б) x:= −5; while X<10 do begin x:=x+2; x:=2*x; end;
   в) i:=0; while i*i <=1.2 do i:=i+0.1;
   г) k:=5; while k /5 <= 2.5 do k:=k+1.5;
   д) t:=100; repeat t:= t/10; until t<=0.1;
   е) x:=0; repeat x:=x+1/10; until sqr(x)>=6/5;
3. Дано натуральное число *N*. Составьте программу, которая выводит на экран все натуральные числа, квадрат которых не больше чем *N*.
4. Составьте программу вычисления значений функции $y = x \cdot \sin x$ для значений x из интервала $[-\pi, \pi]$ с шагом 0.3.

5*. Дано натуральное число *N* и последовательность целых чисел $A_1, A_2, ..., A_N$. Составьте программу, которая последовательно прибавляя члены последовательности, выводит результат, когда сумма впервые превышает число *N*. Если сумма всех членов последовательности не превышает число *N*, выведите об этом сообщение.

Урок 36. Повторение темы «Операторы цикла по условию»

1. Составьте программу вычисления суммы S=0,5+1,5+ +2,5+...+98,5+99,5.

2. Составьте программу вычисления произведения S=1*2+3*4+ +5*6+...+101*102.

3. Составьте программу, которая выводит все делители натурального числа *N*.

4. Составьте программу, которая определяет число разрядов заданного натурального числа *N* (подсказка: сколько раз должна выполняться $N = N$ div 10, чтобы получилось $N = 0$?).

5. Дано натуральное число *N*. Составьте программу, которая выводит все числа от 1 по *N*, у которых последняя цифра кратная 3.

6. Составьте программу, которая выводит все двузначные числа, у которых сумма цифр является четной (подсказка: единичная цифра K1 = K mod 10, десятичная цифра K10 = K div 10).

7. Дана последовательность целых чисел A_1, A_2, ..., A_N. Составьте программу, которая выводит разность произведения элементов с нечетным индексом и сумму элементов с четным индексом.

8*. Даны натуральное число *N* и массив *A*[1..*N*]. Составьте программу, которая создает массив B[1..N], у которого *K*-элемент равен среднему арифметическому первых *K* элементов массива *A* (подсказка: B[K] = (A[1] + A[2] + ... + A[K]) / K).

9*. Дано число *A* больше 1. Составьте программу, находящую наименьшее неотрицательное значение целого *K*, для которого выполнено условие $7^K > A$.

Урок 37. Задания для повторения

1. Составьте программу, которая вычислит значение следующей суммы, пока значение суммы не превосходит заданное число *M*.

$$y = \frac{1}{3} - \frac{1}{10} + \frac{1}{21} - ... + \frac{(-1)^{j+1}}{j \cdot (2 \cdot j + 1)} - ...$$

2. Решите задание 1 двумя спообами: с помощью оператора цикла с параметром и с помощью оператора цикла по условию.

3. Дано натуральное число *N*. Составьте программу, которая вычислит сумму *N* членов последовательности (1/2), (3/4), (5/6), (7/8), ...

4. Дано положительное число *A* и сторона k-квадрата равна $\frac{A}{k}$ ($k = 1,2,...$). Составьте программу, которая определяет, при каком значении *k* суммарная площадь квадратов впервые превосходит A^2.

5. Составьте программу, которая определяет трехзначное число кратное *P*, но не равное ему.

6. Даны натуральное число *N* и массив *A*[1..*N*]. Составьте программу, которая определяет наибольший и наименьший элементы массива.

7. Составьте программу, которая находит сумму цифр натурального числа M.

8*. Даны натуральные числа *B*, *M*, *A*. Последовательность определяется по следующему закону: $Y_1 = B$; $Y_i = \sqrt{M} + A \cdot Y_{i-1}$, i = 2,3,... Составить программу, которая выводит все члены последовательности меньшие, чем $B \cdot M \cdot A$.

9*. Клиент положил в банк *B* сумов. В год сумма сбережений возрастет в *M* процентов. Составьте программу, которая определит, через сколько лет сумма сбережений превысит *A* сумов.

10*. Малое предприятие в первый день выпустило *B* штук товара. Каждый следующий день оно выпускало товара на *M* штук больше, чем в предыдущий день. Составьте программу, которая определит, через сколько дней дневной выпуск товара превысит число *A*.

11*. Вокруг считающего человека по кругу стоят люди в порядке с 1 по *B*. Когда считающий человек доходит до *M* и *M*-человек выходит из круга, счет начинается с единицы. Этот процесс продолжается пока не остается один человек. Составьте программу, которая определяет порядковый номер оставшегося в круге человека.

Урок 38. Работа с символьными и строковыми величинами

На языке Паскаль введены специальные функции и процедуры для работы с символьными и строковыми величинами. На этом уроке мы рассмотрим некоторые из них.

Запись	Задача
Стандартные функции	
Concat(S1,S2,...,SN)	Последовательно объединяет строковые (символьные) константы (переменные):**S1,S2,...,SN**
Length(S)	Определяет длину (количество символов) строки **S**
Pos(b,S)	Производит поиск подстроки **b** в строке **S**
Copy(S,n1,n2)	Копирует подстроку строки **S**, где **n1** — номер символа, с которого выделяется подстрока, **n2** — количество символов в подстроке.
Ord(B)	Определяет код ASCII символа **B**
Chr(a)	Определяет символ кода ASCII, который равен **a**

Стандартные процедуры	
Delete(S,n1,n2)	Удаляет из строки **S n2** символа, начиная с **n1**-символа
Insert (S1,S,n)	Вставляет в строку **S** подстроку **S1,** начиная с позиции **n**
Str(a,S)	Преобразует число **a** в его строковое представление **S**
Val(S,a,c)	Преобразует строковое значение **S** в его численное представление **a**, при этом **c** равняется нулю. Если преобразование невозможно, **a** примет нулевое, **c** – ненулевое значение.

Следует отметить, что значение функции присваивается некоторой переменной, а процедуры записываются без оператора присвоения.

Рассмотрим примеры.

1. Если a = 'В здоровом теле', b = 'здоровый дух'; то после выполнения оператора c:= **Concat**(a, b); значение c равно 'В здоровом теле – здоровый дух'. Аналогичный результат дает оператор присвоения: c:=**a+b**;

2. Если a ='информатика', после выполнения оператора n:= **Length**(a), значение n равняется 11.

3. После выполнения оператора a:=**Pos**('м', 'информатика'), значение a равняется 6; и при выполнении оператора a:=**Pos**('ма', 'информатика'), значение a равняется 6; при выполнении оператора: a:= **Pos**('ка','класс') значение *a* равняется 0; и при выполнении оператора: *a*:= **Pos**('в','класс'), *a* примет значение, равное нулю.

4. После выполнения оператора: a:=**Copy**('информатика',3,5), значение *a* будет равно слову 'форма'.

5. Если a:= 'A', то значение функции **Ord**(a) будет равным 65, т.к. код ASCII (латинской) буквы 'A' равен 65. Если аргумент функции Ord константа, то он пишется в кавычках. Например, Ord('A').

6. Если cod:= 65, то значение функции **Chr**(cod) равно латинской букве 'A', а значение **Chr**(66) равно латинской букве 'B'.

7. Если a:= 'крошка', то после выполнения процедуры **Delete**(a,2,1) получим a = 'кошка'. Полученный результат можно изобразить в виде следующей схемы:

(a:= 'крошка' → Delete(a,2,1) → 'крошка' → a = 'кошка')

8. Если a= 'кошка', a b= 'р', то после выполнения процедуры **Insert**(b,a,2) получим a:= 'крошка', что соответствует следующей схеме: (a= 'крошка', b= 'р'→ Insert(b,a,2); → 'к'+ 'р' + 'ошка' → a:= 'крошка').

9. Если a = 765, то после выполнения процедуры **Str**(a,s), получим: s= '765'.

10. Если s='123', то после выполнения процедуры **Val**(s,a,c) получим a=123 и c= 0; если s= '34BMA5', то после выполнения **Val**(s,a,c) получим a= 0 и c ≠ 0.

Теперь применим функции и процедуры для решения задач.

Пример 1. Составьте программу, которая из заданных слов 'независимое ', 'Узбекистан ', 'государство' составит текст 'Узбекистан – независимое государство '.

Решение. Заданные слова являются константами.

```
Program Tekst;
Const a='независимое '; b='Узбекистан '; c='государство';
Var d : String;
Begin
          d:=Concat(b, '–' a,c); WriteLn(d);
End.
```

Пример 2. Составьте программу, которая перепишет в обратном порядке заданное слово. Например, от слова 'мир' должно получиться 'рим'.

Решение. Обозначим заданное слово через строку *a*, получаемое слово через строку *b*. Строку *b* приравниваем к пустой строке (b:= ''). Определяем длину строки *a*. Копируем каждый символ *a*, начиная слева, прибавляем к *b* слева.

```
Program Obratniy;
Var a, b, belgi: String;  i, len : Integer;
Begin
  Write('Введите слово : '); ReadLn(a);  Len:=Length(a); b:='';
   For i:=1 To Len Do
      begin
        belgi:=Copy(a,i,1);  {копируется i-символ a }
        b:=belgi+b;    {копированный символ a прибавляется к b
        слева}
       end;
  Write(b); readln;
End.
```

Пример 3. Составьте программу, которая определит, содержится ли в заданном слове заданный символ.

```
Program Poisk;
```

```
Var suz : String; belgi : Char;
Begin
  Write('Введите слово: '); ReadLn(suz);
  Write('Введите искомый символ : '); ReadLn(belgi);
   If Pos(belgi,suz)>0 Then WriteLn('ЕСТЬ') Else WriteLn('НЕТ');
End.
```

В результате работы программы на экране появляется одно из слов 'ЕСТЬ' или 'НЕТ'.

Пример 4. Составьте программу, образующую из слов 'скоро' и 'точно' слово 'срочно'.

```
Program Slovo;
Var a,b : String;
Begin
     a:='скоро'; b:='точно';
     Delete(a,2,2);   {получилось a='сро'}
     Delete(b,1,2);   { получилось b='чно'}
      Insert(b,a,4);    {a='срочно'}
      WriteLn(a);
End.
```

Пример 5. Составьте программу, образующую из словосочетаний: «Алишер Навои», «родился», «Великий поэт», «году» и цифры «1441» предложение «Великий поэт Алишер Навои родился в 1441 году».

```
Program Alisher_Navoi;
Const a=' Алишер Навои', b= ' году ', c=1441, d='Великий поэт',
e=' родился в ';
Var yil, s : String;
Begin
        Str(c, yil);  {из числа c=1441 получили строку yil='1441'}
        s:=Concat(d,a,e,yil,b);
WriteLn(s);
End.
```

Пример 6. Составьте программу, которая вычислит сумму цифр заданного целого числа.

```
Program Sifri;
Var son, raqam, len, i, c, natija : Integer; _son, _raqam : String;
Begin  Write('Введите целое число: '); ReadLn(son);
  Str(son,_son);   len:=Length(_son); natija:=0;
  For i:=1 to len Do   begin
              _raqam:=Copy(son, i, 1);  Val(_raqam,raqam,c);
            natija:=natija+raqam;      end;
  WriteLn('Сумма цифр числа', son, '=', natija);
End.
```

Эта программа выдает ошибочный результат, когда число превосходит 32767.

Подумайте о причине ошибки. Внесите изменения в программу так, чтобы можно было получить результат для целых чисел, не превосходящих число 2147483647.

На языке Паскаль строки обладают «свойством массива», т.е. строка считается массивом, а символы строк – элементами массива.

Например, если s — строковая переменная, то s[3] означает 3-символ строки, т.е. если s:='книга', то s[3]= 'и'.

Пример 7. Составьте программу, которая выводит на экран коды ASCII символов заданной строки.

```
Program Kod;
Var s, b : String;   i, L, cod : Integer;
Begin
        Write('Введите строку : '); ReadLn(s);
L := Length(s); {вычислена длина строки и присвоена L }
        For i:=1 to L Do WriteLn(Ord(s[i]));
End.
```

Многие задачи легко решаются с помощью «свойства массива» строк. Пользуясь этим свойством строк, попробуйте решить пример 6.

Вопросы и задания

1. *Для чего предназначена функция Concat? Приведите примеры.*
2. *Можно ли заменить функцию Concat другим действием?*
3. *Какая из функций определяет длину строки? Приведите примеры.*
4. *Какие задачи решает функция Pos?*
5. *В каких случаях значение функции Pos равно нулю? Приведите примеры.*
6. *Для чего применяется функция Copy?*
7. *Может ли значение функции Copy быть числовым? Приведите примеры.*
8. *Какая процедура удаляет часть строки?*
9. *Есть ли возможность на языке Паскаль преобразовать в числовое выражение произвольную символьную или строковую величину? Объясните ответ.*
10. *Расскажите о назначении функций Ord и Chr.*
11. *Расскажите о «свойстве массива» строк.*

Упражнения

1. Определите результат применения функций.
   а) Concat('ма', 'ма');
   б) Concat('да', 'здрав', 'ствует');
   в) a:='море'; Length(a);
   г) Pos('o', 'bahor');
2. Пусть значение строковой переменной S='Информатика'. Определите результат применения следующих действий:

а) Delete(s,5,7);
б) Delete(s,1,2); Delete(s,6,4).

3. Составьте программу, которая определяет возможность получения слово «мать» из введенной строки.

Урок 39. Повторение темы «Работа с символьными и строковыми величинами»

1. Составьте программу, которая выводит на экран заданное слово с добавленными пробелами между символами.

2. Составьте программу, которая определит количество символов «*b*» в заданной строке.

3. Дана строковая линейная таблица A[1..N]. Составьте программу, выводящую на экран те элементы таблицы, которые начинаются с буквы «*м*».

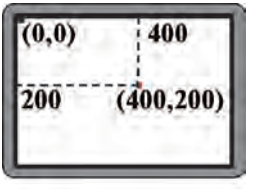
4*. Строка *S* состоит из цифр. Составьте программу, которая образует наибольшее число из цифр строки *S*.

5*. Составьте программу, которая определит возможность получения из строки *A* строку *B* только заменой местами букв.

Урок 40. Перевод экрана в графический режим

Компьютер предназначен для работы не только в текстовом режиме, но и в графическом режиме. В графическом режиме на экране компьютера можно нарисовать различные рисунки. С этой целью в модуль **Graph** (граф) среды Турбо Паскаль включены функции и процедуры для рисования точек, прямых, прямоугольников, окружности и т.д.

Для использования функций и процедур рисования в начале программы следует ввести указание **Uses Graph;**. Это указание предоставит возможность использовать лишь функции и процедуры, содержащиеся в составе модуля. Для того чтобы эти функции и процедуры работали, небходимо перейти в графический режим.



В графическом режиме экран компьютера состоит из мелких точек (пикселей). Как и в текстовом режиме, графический режим имеет **курсор** в виде точки. Изображения получаются из-за передвижения курсора вдоль экрана, ос-

тавляющего за собой видимый или невидимый след. Местоположение курсора на экране определяется его координатами. Начало координат, т.е. точка (0,0), находится в верхнем левом углу экрана. При переходе в графический режим курсор находится в начале координат. Оси координат X и Y направлены, соответственно, вправо и вниз от начала координат, т.е. координаты точек возрастают в этих направлениях. Точка экрана, где находится курсор, называется **текущей точкой**. Число точек на экране будет не более чем 640x480 (0...639x0..479).

Для перевода экрана в графический режим применяется процедура **InitGraph(GD, GM, <путь>);** модуля Graph, где **GD** (GraphDriver) и **GM** (GraphMode) – целочисленные переменные. Их значение зависит от графических возможностей компьютера и выбранного графического режима. Если принять GD:=0; (или GD:=Detect;), то оптимальный графический режим определяется автоматически. <путь> – путь, указывающий местонахождение специального файла с расширением **BGI**, который обеспечивает работу в графическом режиме. В современных компьютерах используется файл EGAVGA.BGI. Если этот файл находится в текущем каталоге, вместо <путь> пишется пустая строка.

Для выхода из графического режима, т.е. чтобы вернутся в текстовый режим, используется оператор **CloseGraph**. Программы, связанные с графикой, имеют следующую структуру:

```
Uses Graph;
Var Gd, Gm : Integer;
        {Описание переменных, соответствующих задаче, связанной с графикой}
Begin
  Gd := 0; {автоматическое определение графического драйвера}
  InitGraph(Gd,Gm, ''); {переход в графический режим}
      {часть для записи решения задачи, связанной с графикой}
  Readln; CloseGraph; {выход из графического режима}
End.
```

В графическом режиме используется 16 цветов. Эти цвета закодированы целыми числами от 0 до 15. В модуле Graph,

как и в текстовом режиме, для каждого цвета выделена константа (см. таблицу из темы работа с экраном). Модуль Graph включает около 80 процедур и функций. Далее рассмотрим некоторые из них.

Процедура **PutPixel(X,Y,Color)** закрашивает точку с координатами (X,Y) в цвет, задаваемый параметром Color. Например, процедура PutPixel(400,200,Red) на экране закрашивает точку с координатами (400,200) красным цветом (см. рисунок выше).

Функция **GetPixel(X,Y)** определяет цвет точки с координатами (X,Y). Например, если Color целочисленная переменная, то оператор Color:= GetPixel(40,50); присваивает переменной Color числовое значение цвета точки (40,50).

Функции **GetMaxX и GetMaxY** определяют максимальные координаты, соответственно, по горизонтали и вертикали. Эти функции полезны для составления программ, не зависящих от графического адаптера компьютера и графического режима.

Пример 1. Составьте программу, которая с помощью точек вычерчивает отрезок красного цвета, соединяющий точки (0,0) и (639,479).

Решение. Определим значение следующих целочисленных переменных: bx:=0; ox:= GetMaxX; (можно и так: ox:=639;). Легко проверить, что на отрезке [bx, ox] начальное значение целочисленной линейной функции y=[GetMaxY·x/GetMaxX] равно 0, а конечное значение равно 479 (вместо GetMaxY можно написать 479).

Следующая программа, составленная с помощью оператора цикла с параметром, решит задачу:

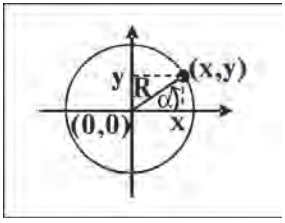
```
Uses Graph;
Var   gd, gm:integer;
      bx, y, ox: Integer; x: LongInt;
Begin
  Gd := 0; InitGraph(Gd, Gm, '');
  bx:=0; ox:= GetMaxX;
For x:= bx to ox do                    begin
y:= trunc(GetMaxY*x/GetMaxX); putpixel(x, y, red);      end;
  Readln; CloseGraph;
End.
```

Подумайте: почему тип переменной x определен как LongInt!

Пример 2. Составьте программу, которая с помощью точек вычерчивает окружность желтого цвета в центре экрана.

Решение. Введем целочисленные переменные для координат центра экрана: markazX:=trunc(GetMaxX/2); markazY:=trunc(GetMaxY/2);

(или markazX:=639; и markazY:=479). По определению $\sin\alpha = \frac{y}{R}$ и $\cos\alpha = \frac{x}{R}$. Следовательно, точки окружности с центром в точке (0,0) и радиусом R можно определить с помощью формул: x = r·cos(α), y = r·sin(α), где угол принимает значения от 0 до 2 π. Чтобы точки окружности расположились плотнее, изменение шага угла α положим 0,01. Учитывая сказанное, составим следующую программу:




```
Uses Graph;
Var   gd, gm:integer;
    x, y, markazX, markazY: Integer; R, alfa: real;
Begin
  Gd := 0; InitGraph(Gd, Gm, '');
  Write('R= '); readln(R);
  markazX:=trunc(GetMaxX/2); markazY:=trunc(GetMaxY/2); alfa:=0;
while alfa<=2*pi do          begin
    x:= markazX +trunc(R*cos(alfa));
    y:= markazY +trunc(R*sin(alfa));
    putpixel(x, y, 14); alfa:= alfa+0.01;
                                 end;
  Readln; CloseGraph;
End.
```

Пример 3. Составьте программу, которая с помощью точек вычерчивает зеленым цветом график функции $y=x^2$ на отрезке $x \in [-5, 5]$, с вершиной в центре экрана.

Решение. Программа этой задачи следующая:

```
uses Graph;
var   gD, gm: Integer;
    markazX, markazY: integer; x, y: real;
begin
     gD := 0; InitGraph(gD, gm, '');
      markazX:=trunc(getmaxx/2); markazY:=trunc(getmaxy/2);
x:= −5;
    while x<=5 do   begin
   y:=x*x; putpixel(trunc(10*x+markazX), trunc(−5*y+markazY),blue);
      x:=x+0.01;      end;
    Readln;     CloseGraph;
End.
```

В программе, чтобы увеличить масштаб, переменная *x* умножена на 10, *а y* на 5. Ветви параболы направлены снизу верх за счет знака «−».

Попробуйте самостоятельно увеличить масштаб и рассмотрите случай без знака минус.

Вопросы и задания

1. *Для каких целей используется модуль Graph?*
2. *Чем определяется местоположение точки на экране?*
3. *С помощью какого указания экран переводится в графический режим?*
4. *Какая процедура служит для выхода из графического режима?*
5. *Расскажите о функциях GetPixel, GetMaxX и GetMaxY.*

Упражнения

1. Составьте программу, которая переводит экран в графический режим и при нажатии клавиши Enter возвращает в текстовый режим.

2. Составьте программу, которая выводит в четырех углах экрана по одной желтой точке.

3. С помощью точек нарисуйте горизонтальную линию, разделяющую экран на две равные части.

4. Выведите на экран разноцветные точки с помощью функции Random.

5. Составьте программу, которая с помощью точек вычерчивает график функции $y=3x+5$ на отрезке $x \in [-10, 10]$.

Урок 41. Повторение темы «Перевод экрана в графический режим»

1. Составьте программу, которая рисует прямоугольник с разноцветными сторонами.

2. Составьте программу, которая определяет принадлежность точки, заданной графическими координатами, к прямой линии, заданной через графические координаты, через которые она проходит (подсказка: использовать функцию GetPixel для определения цвета точки и сравнить с цветом прямой).

3. Составьте программу, которая нарисует на экране 15 параллельных отрезков (подсказка: использовать оператора цикла для рисования и подсчета отрезков).

4. Составьте программу, которая нарисует картину «звездного неба» с помощью функции Random.

5. Составить программу, которая рисует оси координат, отражает название осей в соответствующем месте, рисует график функции $y=|x|$ на отрезке $x \in [-7, 7]$.

6*. Составьте программу, которая рисует 7 вложенных друг в друга окружностей (подсказка: использовать оператор цикла для определения радиусов).

7*. Составить программу, которая нарисует окружность, мерцающую 7 раз (подсказка: нарисуйте окружность, а для замедления процесса выполните «пустой» цикл, нарисуйте окружность с цветом фона, а для замедления процесса выполните «пустой» цикл, повторите эти действия 7 раз).

Урок 42. Возможности рисования фигур на языке Паскаль

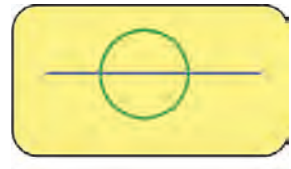
Вы уже видели как, с помощью точек, хоть и не легко, можно рисовать различные фигуры. Но язык Паскаль располагает процедурами для рисования готовых фигур. Эти процедуры рисуют фигуры на основе заранее определенных цветов. Для определения цвета линии применяется процедура **SetColor(rang)**, а для определения цвета фона – процедура **SetBkColor(rang)**. Здесь **«rang»** — целочисленная переменная или константа, определяющая код цвета или имя константы. Выбранный цвет называется текущим. Если заранее цвет не задан, то текущим считается белый цвет.

Процедура **Line(X1,Y1,X2,Y2)** рисует отрезок текущего цвета, соединяющий точки с координатами (X1,Y1) и (X2,Y2).

Процедура **Circle(X,Y,R)** рисует окружность с радиусом R текущего цвета с центром в точке (X,Y). Теперь легко можно решить задачи, связанные с окружностями.

Пример 1. Нарисуйте отрезок синего цвета с вершинами в точках (10,200) и (630,200) и окружность радиуса 100 зеленого цвета с центром в точке (300,200). Фон заполнить желтым цветом.

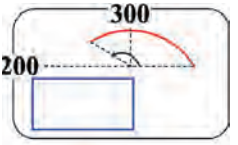
```
Uses Graph;
Var gd, gm : Integer;
Begin  Gd:=Detect; InitGraph(gd, gm,'');
    SetBkColor(Yellow);
    Setcolor(Blue);     Line(10,200,630,200);
    Setcolor(Green);    Circle(300,200,100);
Readln;     CloseGraph;
End.
```



Процедура **Ellipse(X,Y,BB,OB,XR,YR)** рисует дугу (эллипса) текущего цвета от угла *BB* до угла *OB* с центром в точке (*X,Y*), с радиусами по оси *x* и *y*, соответственно, равными *XR* и *YR*. Углы задаются градусами. Если *XR=YR* , то рисует дугу окружности.

Процедура **Rectangle(X1,Y1,X2,Y2)** рисует прямоугольник текущего цвета, с верхним левым углом в точке (X1,Y1) и правым нижним углом в точке (X2,Y2).

Пример 2. Нарисуйте дугу красного цвета от угла 0° до угла 135° с центром в точке (300,200), с радиусами по оси *x* и *y*, соответственно, равными 100 и 50 и прямоугольник синего цвета с верхним левым углом в точке (10, 220) и правым нижним углом в точке (300, 400).




```
Uses Graph;
Var gd, gm : Integer;
Begin
     Gd:=Detect; InitGraph(gd,gm, ''); Setcolor(4);
    Ellipse(300,200,0,135,100,50);
  Setcolor(1);  Rectangle(10,220,300,400);
     Readln;     CloseGraph;
End.
```

Процедура **DrawPoly(BS,KM)** рисует ломаную линию, где BS – количество точек перелома линии, KM — имя массива, содержащего координаты точек перелома. Если координаты начальной и конечной точки ломаной совпадают, то получится многоугольник.

На языке Паскаль имеются также процедуры, рисующие геометрические фигуры, окрашенные различным способом. Цвет контура этих фигур устанавливается процедурой **SetColor.** Цвет для заливки и способ заливки устанавливаются процедурой **SetFillStyle(metod,color)**;, где **metod** — способ заливки, **color** — цвет заливки.

На языке Паскаль фигуры можно заполнить выбранным цветом различными способами. Способы заливки, как и цвета, закодированы целыми числами. В модуле Graph для каждого способа заливки выделена константа. Следующая таблица отражает способы заливки, их коды и соответствующие константы:

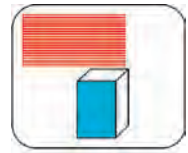
Способы заливки	**Коды**	**Имя константы**
Заливка цветом фона	0	EmptyFill
Заливка заданным цветом	1	SolidFill
Заполнение толстой горизонтальной линией	2	LineFill
Заполнение тонкими наклонными влево линиями	3	LtSlashFill
Заполнение толстыми наклонными влево линиями	4	SlashFill
Заполнение толстыми наклонными вправо линиями	5	BkSlashFill

Заполнение тонкими наклонными вправо линиями	6	LtBkSlashFill
Заполнение прямоугольными клеточками	7	HatchFill
Заполнение тонкими наклонными клеточками	8	XHatchFill
Заполнение толстыми наклонными линиями	9	InterLeaveFill
Заполнение редкими точками	10	WideDotFill
Заполнение частыми точками	11	CloseDotFill
Узор устанавливается пользователем	12	UserFill

Процедура **Bar(X1,Y1,X2,Y2)** рисует окрашенный прямоугольник с верхним левым углом в точке (X1, Y1) и правым нижним углом в точке (X2, Y2), используя текущий цвет и метод заливки.

Процедура **Bar3D(X1,Y1,X2,Y2,a,b)** рисует окрашенный параллелепипед, используя текущий цвет и метод заливки. Здесь **a** — длина боковой стороны параллелепипеда, а **b** – логическое выражение, и если его значение «истина», то вычерчивается правая верхняя грань, если «ложь» — не вычерчивается.

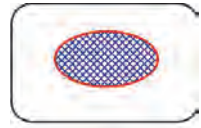
Пример 3. Нарисуйте прямоугольник, заполненный толстыми горизонтальными линиями и голубой параллелепипед.



```
Uses Graph;
Var gd, gm : Integer;
Begin
    Gd:=Detect; InitGraph(gd,gm, '');
    SetFillStyle(2, 4);    Bar(10,10,400,200);
    SetFillStyle(1,9);    Bar3D(100,200,350,400,50,True);
    Readln;     CloseGraph;
End.
```

Процедура **FillEllipse(X,Y,XR,YR)** рисует окрашенный эллипс с центром в точке (X,Y), радиусами XR (по горизонтали) и YR (по вертикали), используя текущий цвет и метод заливки.

Пример 4. Нарисуйте эллипс с красным контуром, заполненный тонкими наклонными клеточками.



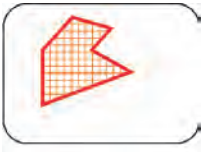
```
Uses Graph;
Var gd, gm : Integer;
Begin
   Gd:=Detect; InitGraph(gd,gm, ''); SetColor(Red);
   SetFillStyle(8,1);  FillEllips(320,240,200,100);
    Readln;     CloseGraph;
End.
```

Процедура **FillPoly(BS,KM)** рисует многоугольник, используя текущий цвет и метод заливки, где BS — количество углов многоугольника, KM — имя массива, содержащего координаты точек вершин

многоугольника. В отличие от процедуры DrawPoly автоматически соединяет начальные и конечные вершины многоугольника.

Пример 5. Нарисуйте шестиугольник красным контуром, заполненный красными прямоугольными клеточками. Вершины шестиугольника задаются в массиве KM.

Решение. Применим процедуру рисования ломаной линии с 7 вершинами. Координаты 7-ой вершины и 1-ой вершины зададим одинаковыми, т.е. получим шестиугольник.



```
Uses Graph;
Const bs=6;
Var  gd, gm : Integer;
     km : Array[1..bs,1..2] of Integer;
Begin   gd:=0; InitGraph(gd,gm, '');  Setcolor(4);
  SetFillStyle(7,4); {выбор способа и цвета}
   km[1,1]:=300;      km[1,2]:=10;
   km[4,1]:=400;      km[4,2]:=190;
   km[2,1]:=200;      km[2,2]:=80;     km[5,1]:=300;   km[5,2]:=80;
   km[3,1]:=200;      km[3,2]:=200;    km[6,1]:=400;   km[6,2]:=40;
 FillPoly(bs,km); {если написать DrawPoly(bs,km); получится неокрашенный шестиугольник}
End.
```

Вопросы и задания

1. *Какая процедура устанавливает цвет рисования?*
2. *Покажите на практике возможности рисования отрезка языка Паскаль.*
3. *Какая процедура рисует окружность?*
4. *Какие фигуры можно рисовать с помощью процедуры Ellipse?*
5. *Что означают x1, y1, x2 и y2 в операторе рисования прямоугольника?*
6. *Какие фигуры можно рисовать с помощью процедуры DrawPoly?*
7. *Продемонстрируйте изменение цвета фона на практике.*
8. *Какие фигуры можно рисовать с помощью процедуры SetFillStyle?*
9. *Расскажите о способах рисования окрашенных и неокрашенных многоугольников.*

Упражнения

1. Составьте программу, которая рисует горизонтальную и вертикальную прямые, проходящие через центр экрана.

2. Нарисуйте в центре экрана 4 окружности желтого цвета с радиусами меньше 100.

3. Заполните экран желтыми горизонтальными линиями.

4. Составьте программу, которая рисует светофор.

5. Составьте программу, которая рисует правильный пятиугольник красного цвета.

Урок 43. Повторение урока «Возможности рисования фигур на языке Паскаль»

1. Составьте программу, которая рисует флаг Узбекистана.

2. Нарисуйте в четырех углах экрана прямоугольники красного цвета размером 60x40.

3. Разделите экран на 4 равные части и закрасьте их красным, желтым, зеленым и синим цветами.

4. Нарисуйте в центре экрана круг желтого цвета радиусом 100.

5. Нарисуйте вид звездного неба с луной.

6. Составьте программу, которая нарисует лучезарное Солнце над морем. Для рисования моря используйте процедуру, рисующую дугу.

7. Нарисуйте шестиугольник, заполненный редкими красными точками.

8*. Нарисуйте 12 квадратов размерами 40x40, демонстрирующих 12 способов заливки.

9*. Составьте прграмму, которая нарисует светофор с последовательно загорающимися лампочками.

Урок 44. Работа с файлами

На предыдущих уроках вы ознакомились с операторами ввода и вывода данных. Оператор ввода предоставит возможность ввода данных клавиатурой, а оператор вывода выводит информацию на экран. Встречаются задачи, в которых выводимая информация не помещается на экране.

А результаты некоторых задач требуется сохранить для дальнейшего пользования.

В таких случаях целесообразно сохранить информацию в виде файла. Вам известно о файлах и об их форматах (текстовые, графические и т.д.). Язык Паскаль предоставляет возможность работать с файлами разных форматов.

На языке Паскаль для этого введены специальные **переменные типа файл** (файловые переменные). Через файловые переменные связывают файлы, находящиеся во внешней памяти. Так как значение числовых переменных —

число, значение строковых переменных — строка, то значение файловых переменных это файл. Файловые переменные тоже должны быть описаны. Далее мы рассмотрим только текстовые файлы. Текстовые файлы состоят из строк различной длины. Длина строки не должна превышать 255 символов. Для описания файловых переменных используется служебное слово **text**.

Например,

var ftext : Text; {ftext — текстовой файл}

Для работы с файлами следует придерживаться следующих действий:

1. Связать файловую переменную с конкретным файлом во внешней памяти.
2. Открыть файл для «чтения» или «записи».
3. Чтение информации из файла или ее запись в файл.
4. Закрыть файл.

Файловая переменная связывается с файлом во внешней памяти с помощью процедуры **Assign(f, <имя файла>);** где f — файловая переменная; <**имя файла**> — строковая переменная или константа, выражающая имя файла во внешней памяти.

Если файл не находится в текущем каталоге, то указывается его **полное имя**.

Например, если файл «Навруз.txt» находится в каталоге «Праздники» диска «D», то его полное имя записывается в следующем виде:

d:\Праздники\Навруз.txt

Чтобы связать этот файл с файловой переменной f, процедура Assign записывается в следующем виде:

Assign(f, 'd:\Праздники\Навруз.txt');

Процедура Assign определяет значение файловой переменной, т.е. определенный файл во внешней памяти. Чтобы пользоваться файлом (чтение информации из файла или запись информации в файл) этот файл необходимо «открыть». Текстовые файлы нельзя открыть для чтения и для записи одновременно.

Из файла, открытого для чтения, можно только считывать информацию. А в файл, открытый для записи, можно только записать информацию.

На языке Паскаль файлы «для записи» можно открыть следующими двумя способами:

1. Создать новый файл и открыть его для записи.

2. Открыть существующий файл для добавления информации.

Процедура **Rewrite(f)** создает новый файл во внешней памяти и открывает его для записи. Он должен быть связан с некоторым файлом с помощью процедуры Assign.

Например, после выполнения процедур Assign(f, 'sumalak.txt'); Rewrite(f); в текущем каталоге создается новый файл под именем «sumalak.txt» и открывается для записи. Если в текущем каталоге до этого находился файл с таким именем, то теперь он стирается и вместо него записывается новый файл.

После использования открытый файл следует закрыть. Это осуществляется с помощью процедуры **Close(f)**. Эта процедура закрывает открытый (для чтения, для записи) файл. До применения процедуры Close, файл из внешней памяти, соответствующий файловой переменной, обязательно должен быть открыт.

Для записи в файл информации (для ввода информации в файл) используются следующие операторы:

Write(f, <список данных>); или
WriteLn(f, <список данных >);

где **f** — файловая переменная; <список данных> — последовательность переменных или констант, разделенных между собой запятыми. Эти операторы записывают в файл переменные или константы, указанные в списке данных.

Пример 1. Составьте программу, которая в текущем каталоге создает файл с именем «Gimn.txt» и записывает в отдельные строки первые 4 строки гимна Узбекистана.

Решение. Для записи вводимой информации в отдельных строках используем оператор WriteLn.

```
Program Gimn1;
Var _ gimn: Text;      satr : String; m:integer;
Begin
 Assign(_gimn, 'Gimn.txt');  Rewrite(_gimn);
 For m:=1 to 4 do            begin
 Write('Введите ', m, '-строку гимна: '); ReadLn(satr);
 WriteLn(_gimn,  satr);     end;
```

```
 Close(_gimn);
End.
```

Мы рассмотрели создание нового файла во внешней памяти. Иногда приходится продолжить находящийся во внешней памяти текстовый файл, т.е. в него требуется добавить новую информацию. В этом случае вместо процедуры Rewrite применяется процедура **Append(f).** Эта процедура открывает указанный файл во внешней памяти «для записи». Если такой файл не существует во внешней памяти, то произойдет ошибка.

Следовательно, открываемый файл с помощью процедуры Append должен существовать во внешней памяти.

Пример 2. Составьте программу, которая открывает файл «Gimn.txt» из примера 1 и записывает припев гимна Узбекистана в продолжении.

Решение: Используем процедуру Append для продолжения файла.

```
Program Gimn2;
Var  f : Text;        naqorat: String; m:integer;
Begin
 Assign(f, 'Gimn.txt'); Append(f);
   For m:=1 to 4 do          begin
 Write('Введите ', m, ' -строку припева: '); ReadLn(naqorat);
 WriteLn(f, naqorat);        end;
 Close(f);
End.
```

Процедура **Reset(f)** открывает файлы для чтения. Она должна быть связана с некоторым файлом с помощью процедуры Assign. Если такой файл не существует во внешней памяти, то произойдет ошибка.

Из файла, открытого для чтения, данные считываются с помощью следующих операторов:

Read(f, <список переменных>); или
ReadLn(f, <список переменных>);

Здесь **f** — файловая переменная; <**список переменных**> — последовательность переменных, разделенных между собой запятыми. Эти операторы считывают из файла значения переменных, указанные в списке данных.

Оператор Read применяется, в основном, если в файле записаны числовые данные, т.е. когда строка файла состоит из чисел, разделенных пробелом. Оператор Read считывает каждое число по отдельности. По окончанию данных в строке, он переходит на следующую строку.

Пример 3. Составьте программу, которая вычислит площадь треугольника, стороны которого записаны в файле «uchbur.in». Файл «uchbur.in» содержит одну строку, в которой числа, являющиеся значениями сторон, записаны через пробел.

Решение. Для чтения данных из файла применяется оператор Read. Площадь треугольника вычисляется по формуле Герона.

```
Program Ploshad;
Var f : Text;       a, b, c, yp, s : Real;
Begin
        Assign(f, 'uchbur.in');
        Reset(f);    {файл «uchbur.in» открыть для чтения}
        Read(f, a); Read(f, b); Read(f, c);
        {значения a,b,c считывались из файла «uchbur.in»}
        Close(f);    {закрыть файл «uchbur.in»}
        yp:=(a+b+c)/2; s:=sqr(yp*(yp−a)*(yp−b)*(yp−c));
        WriteLn('Площадь треугольника=', s);
End.
```

Оператор ReadLn считывает строку целиком. В текстовых файлах данные можно считывать по строкам. Например, чтобы считывать 10 строку файла, обязательно следует считывать первые 9 строк.

Пример 4. В файле «klass.txt» (в каждой строке фамилия одного ученика) записан список учеников 9 класса. Составьте программу, которая выводит фамилию 12 учеников на экран.

Решение. Фамилии 12 учеников записаны в 12 строке файла «klass.txt». Чтобы считывать 12 строку файла, необходимо считывать первые 11 строк.

Для этого используется оператор цикла с параметром.

```
Program Klass;
var fio : Text;     i : Integer; fam : String;
Begin
            Assign(fio, 'klass.txt'); Reset(fio);
            For i:=1 to 11 Do ReadLn(fio, fam);
            ReadLn(fio, fam);  Close(fio);
            WriteLn('12-o‘quvchining familiyasi:', fam);
End.
```

Если в рассмотренном примере потребуется вывести фамилии всех учеников из файла «klass.txt», возникнет проблема. Потому что неизвестно, сколько строк в файле.

В таких случаях применяется функция **Eof(f)**. Функция Eof логическая, т.е. если в файле не останется строк для считывания, она принимает значение «Истина», в противном случае – значение «Ложь».

Пример 5. В файле «klass.txt» записан список учеников 9 класса. Составьте программу, которая выводит фамилии всех учеников на экран.

Решение. Используется оператор цикла с условием While.

```
Program Sinf;
var fio : Text;     fam : String;
```

```
Begin
        Assign(fio, 'klass.txt'); Reset(fio);
        While Not(eof(fio)) Do  begin
        ReadLn(fio, fam);
        WriteLn(fam);              end;
        Close(fio);
End.
```

Повторение в программе продолжается до тех пор, пока условие Not(eof(fio)) не примет значение «ложь», т.е. пока функция eof(fio) не примет значение «истина». Как только функция Eof(fio) принимает значение «истина», т.е. не останется данных для чтения в файле «klass.txt», повторение прекратится.

По мере надобности в одной программе можно открыть несколько файлов. Если они открываются и закрываются по очереди, т.е. после закрытия одного открывается следующий, то можно использовать одну файловую переменную.

В противном случае для каждого файла нужно описывать отдельную файловую переменную.

Вопросы и задания

1. *Что понимается под файловой переменной?*
2. *Какая процедура связывает файловую переменную с файлом из памяти?*
3. *Расскажите о назначении процедуры Rewrite.*
4. *Какая процедура применяется для закрытия файла?*
5. *Какие операторы записывают информацию в текстовой файл?*
6. *Какая процедура применяется для добавления новой информации в файл?*
7. *Какая процедура открывает файл для чтения?*
8. *Какие операторы применяются для чтения информации из текстового файла?*
9. *Укажите разницу в работе операторах Read и ReadLn.*
10. *Если в операторе ReadLn участвуют несколько переменных, то как они записываются?*
11. *Расскажите о назначении функции Eof.*

Упражнения

1. Составьте программу, которая сохраняет в текстовом файле «NEDELYA.TXT» вводимые с клавиатуры названия дней недели.

2. Составьте программу, которая добавит в продолжении файла «NEDELYA.TXT» дни недели на английском языке.

3. Составьте программу, которая выводит на экран названия дней недели, считывая их из файла «NEDELYA.TXT».

Урок 45. Повторение темы «Работа с файлами»

1. Составьте программу, которая создает текстовый файл «KLASS.TXT», содержащий сведения о фамилиях и именах ваших одноклассников.

2. Файл «klass.txt» содержит фамилии учеников 9-класса. Составьте программу, которая выводит фамилии учеников, начинающиеся с буквы «М» из файла «klass.txt».

3. Файл «klass.txt» содержит фамилии учеников 9-класса. Составьте программу, которая из файла «klass.txt» отбирает фамилии учеников, начинающиеся с буквы «В», и записывает их в файл «vklass.txt».

4*. Составьте программу, которая вычисляет значения функции $y = \sin^2 x$ в интервале $[-\pi, \pi]$ с шагом 0,01 и результаты сохраняет в файле «sinus.out».

5*. Составьте программу, которая добавит объяснение в продолжение файла «sinus.out».

Урок 46. Процедуры и функции

Нередко одну и ту же последовательность действий приходится повторять в различных частях программы. На языке Паскаль часто повторяющиеся действия можно отделить от основной программы и организовать из них отдельные блоки, так называемые **процедуры** и **функции**.

Каждой составленной процедуре и функции необходимо присвоить **имя**. Потому что обращение к конкретной процедуре или функции осуществляется через ее имя. Программа, в которой эффективно использованы процедуры и функции, станет более простой и понятной.

Если процедуры и функции состоят из множества операций, чем они отличаются?

Функции составляются для вычисления некоторого значения, и в итоге результат вычисления присваивается именем функции. Процедуры составляются для выполнения определенной последовательности действий.

Здесь важен не результат вычислений, а выполнение последовательности действий.

Например, если необходимо вычислить площадь треугольника, с заданной длиной сторон, то лучше использовать **функцию**. Потому что, каждый раз вычисляется одно значение — площадь треугольника. Если требуется нари-

совать несколько треугольников, то нужно составить процедуру, организующую процесс рисования треугольников. Процедуры и функции можно составить так, что их выполнение будет зависеть от некоторых значений, которые передаются им непосредственно перед вызовом. Эти значения называются **параметрами**, а таким образом составленные процедуры и функции — **процедуры и функции с параметрами**.

Процедуры и функции начинаются с заголовка. Общий вид заголовка процедуры приводится ниже:

Procedure <имя процедуры > (параметры);

Заголовок функции имеет вид:

Function <имя функции> (параметры) : <тип значения функции>;

Если процедура (функция) с параметрами, то в заголовке описываются типы параметров.

Например, Function stepen(a, п : Integer):Integer либо Procedure shifr (stroka:String);

Структура процедуры:

```
Procedure <имя процедуры >(параметры);
Label
    <метки>;
Const
   <описание констант>;
Var
  <описание переменных>;
«Внутренние» процедуры и функции;
begin
  <тело процедуры>  {программа}
end;
```

Структура функции:

```
Function <название функции>(параметры);
Label
   <метки>;
Const
  <описание констант>;
Var
<описание переменных>
«Внутренние» процедуры и функции;
begin
  <тело функции>  {программа}
end;
```

Как видите, процедура и функция имеют похожую структуру и почти не отличаются от структуры программы. Следует заметить, что константы и переменные, описанные в основной программе, называются **общими (глобальными).**

Они доступны в любой части программы, в том числе и внутри процедур и функций. Константы и переменные,

описанные внутри конкретной процедуры (функции), называются **локальными (местными)**, и доступны они только в пределах этой процедуры (функции). На языке Паскаль глобальная и локальная переменные могут иметь одно и тоже имя.

В этом случае, внутри процедуры (функции), где описана эта переменная, принимается значение локальной переменной, а в остальных частях программы – значение глобальной переменной.

Пример 1. Составьте программу, которая будет находить сумму всех целых чисел в промежутках [20, 83], [178, 391], [211, 746].

Решение. Для вычисления суммы трех промежутков приходится использовать три раза оператор цикла с параметром. Приняв начальные и конечные значения промежутков как параметр, и используя функцию для вычисления суммы всех целых чисел промежутка, можно решить задачу.

```
Program Summa;
Var  i, s, s1, s2, s3 : Integer;
Function Sum(n1, n2:Integer):Integer;
Begin
        s:=0;   For i:=n1 To n2 Do s:=s+i;   Sum:=s;
End;
Begin
        S1:= Sum(20,83); s2 := Sum(178,391); s3:=Sum(211,746);
        S:=s1+ s2 + s3;   WriteLn('S= ', s);
End.
```

Пример 2. Составьте программу, которая по заданным значениям сторон треугольника вычисляет его высоту.

```
Program Treugolnik;
Var a, b, c, ha, hb, hc: real;
Function  H_UB(a, b, c: real): real;   { a, b, c — Стороны треугольника }
  Var p, s: real;
Begin
   p:= (a+b+c)/2;          { полупериметр}
   s:= Sqrt(p*(p-a)*(p-b)*(p-c));        {Площадь}
   H_UB:= 2*s/a;         {Функции присвоено значение}
End;
Begin
    Write('Введите стороны треугольника ( a,b,c) '); Readln(a,b,c);
    ha:= H_UB(a, b, c);   hb:= H_UB(b, a, c);   hc:= H_UB(c, b, a);
    Writeln('Высота треугольника:');
```

```
    Writeln('ha=',ha:10:4, 'hb=',hb:10:4, 'hc=',hc:10:4);
     Readln;
End.
```

Пример 3. Даны три треугольника со следующими координатами своих вершин:

1) (120,20), (80,170), (140,150);
2) (200,97), (500,156), (210,180);
3) (300,190), (200,390), (415,222).

Составьте программу, которая рисует эти треугольники красным, желтым и зеленым цветами, соответственно.

```
Uses Graph;
      var  gd, gm:Integer;
 Procedure Treugolnik(x1,y1,x2,y2,x3,y3,col:Integer);
 begin
 SetColor(col);   Line(x1,y1,x2,y2);   Line(x2,y2,x3,y3);
  Line(x3,y3,x1,y1);
end;
   Begin gd:=0;  InitGraph(gd,gm, '');
   Treugolnik(120,20,80,170,140,150,4);
   Treugolnik(200,97,500,156,210,180,14);
   Treugolnik(300,190,200,390,415,222,2); ReadLn;
CloseGraph;
End.
```

Анализ приведенных примеров позволяет сделать следующее заключение: при обращении к процедуре ее имя пишется отдельно, в то время как имя функции пишется в составе какого либо оператора.

Вопросы и задания

1. *С какой целью включают в программу процедуры и функции?*
2. *Чем отличается функция от процедуры?*
3. *Расскажите о процедурах и функциях с параметрами.*
4. *Как выглядит структура процедуры?*
5. *Как выглядит структура функции?*
6. *Расскажите о локальных и глобальных переменных.*
7. *В каких случаях вместо функции можно использовать процедуру?*

Упражнения

1. Даны три прямоугольника, у каждого из которых известны координаты вершин одной диагонали: 1) 20, 20 и 80, 200; 2) 200, 97 и 500, 156; 3) 300, 120 и 400, 420. Составьте программу, которая рисует

эти прямоугольники красным, желтым и зеленым цветами, соответственно.

2. Составьте программу, которая с помощью процедуры сумеет вычислить сумму $S=1\cdot5+2\cdot6+3\cdot7+...+n\cdot(n+4)$ для заданного натурального числа n.

3. Составьте программу, которая определит наибольшее из трех чисел. Для этого составьте функцию, определяющую наибольшее из двух чисел, и воспользуйтесь ею.

Урок 47. Повторение темы «Процедуры и функции»

1. Составьте программу, которая с помощью процедуры заменит в тексте символ 'a' с 'g', символ 'm' с 's', символ 'f' с 'h'.

2. Составьте программу, которая вычислит значение функции $y = x^5 + 3x$ для следующих значений x: −9, −5, −2, 2, 5, 7. Составьте функцию для вычисления степени с помощью произведения.

3. Составьте программу, которая с помощью процедуры нарисует 15 вложенных окружностей.

Уроки 48—49. Задания для повторения

1. Составьте программу, которая нарисует квадрат со стороной a и окружность, вписанную в квадрат.

2. Составьте программу, которая нарисует окружность с радиусом R и вписанным в нее квадратом. Значение радиуса R ввести через клавиатуру.

3. Составьте программу, которая на рисует прямоугольник со сторонами a и b, и вписанным в него эллипсом. Значение a и b ввести через клавиатуры.

4. Заполнить экран синим фоном с вертикальными белыми прямыми. Расстояние между прямыми − 20 пикселей.

5. Составьте программу, которая нарисует 15 кругов разного цвета.

6. Составьте программу, которая нарисует стол.

7. Составьте программу, которая нарисует дом.

ГЛАВА III СОЗДАНИЕ WEB-СТРАНИЦ

Урок 50. Понятие о HTML

Служба WWW Интернета зависит, в основном, от веб-сайтов и веб-страниц, поэтому естественен вопрос: «Как создаются веб-страницы?». Веб-страницы создаются специальными программами. Например, с помощью XML, HTML, редакторов как Microsoft FrontPage, Macromedia Home Site, Adobe Dreamweaver, серверных скриптов (языки сценариев) как PHP, JavaScript и другие. Все эти программы основываются на языке HTML (Hypertext Markup Language — язык маркировки гипертекстов).

Язык HTML не является программным языком. Для создания на этом языке документов (веб-страниц) достаточно такого, например, текстового редактора, как «Блокнот» системы Windows.

Команды языка HTML записываются между знаками «<» и «>» и называются **дескрипторами** (англ. описывающий) или **тэг** (англ. tag – метка, ярлык, признак).

Например, запись <**HTML**> означает начало документа языка HTML. Тэги записываются буквами латинского алфавита, но записи букв в верхнем и нижнем регистрах будут восприняты одинаково, т.е. тэги <HTML> и <html> одинаковые. В общем, тэги делятся на два типа:

1. Парные тэги (контейнер-тэги): на каждого открывающего тэга вида <B> закрывающий тэг вида </B>, первый из них означает начало некоего действия, а второй – окончание этого действия.

2. Непарные тэги: тэг открывается в виде <D>, не обязательно его закрывать, например, как тэг
, переводящий записанный после себя текст на новую строку.

HTML-документ – это просто текстовый файл с расширением **«html»** или **«htm»**, записанный на простом текстовом редакторе. HTML-документ записывается между тэгами <HTML> и </HTML>. Если HTML-документ загрузить

в память, то он отражается на экране с помощью браузера в виде веб-страницы.

HTML-документ, в основном, состоит из двух разделов. Первый из них – раздел HEAD (англ. головная часть или заголовок) заключается между тэгами <HEAD> и </HEAD>.

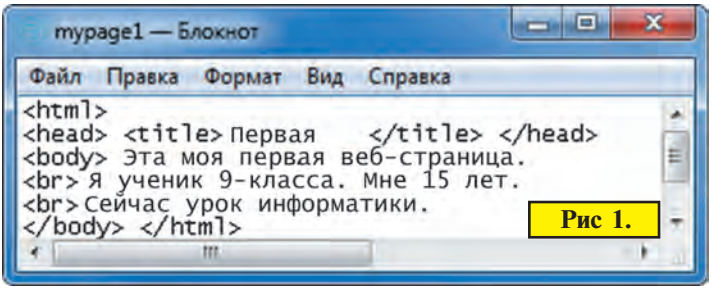
Второй раздел – **BODY** (тело) заключенный между тэгами <BODY> и </BODY>, отражает содержание документа. Если HTML-документ должен отражать фрейм-структуру (информация отражается в окне браузера в отдельных областях), то вместо раздела BODY применяется раздел **FRAMESET** (FRAME SET – строение структур, с помощью парного тэга <FRAMESET>). Обычно, в HTML-документе рекомендуют написать парные тэги <HEAD> и <BODY>, но это не обязательно.

Необходимый элемент веб-страницы — это название документа, которое вводится парным тэгом **<TITLE>**. На веб-странице он используется один раз. Название веб-страницы отражается на строке заголовка браузера и не отражается внутри веб-страницы. И поэтому его можно написать в любом месте (обычно в разделе HEAD) веб-страницы. Веб-странице можно задать любое название, для примера, свое имя. Язык HTML непрерывно развивается. В свою очередь развиваются и браузеры. В настоящее время для создания веб-страниц используют, в основном, язык HTML-4. Его команды не смогут выполнить «старые» браузеры (Internet Explorer-4 и Internet Explorer-6). Известно, что браузеры Internet Explorer, Opera, FireFox, Mozilla и Netscape отличаются. Следовательно, браузеры могут отражать HTML-документ с некоторой разницей.

Самая простая веб-страница состоит только из текста. Мы тоже начнем создание веб-страницы с ввода текста. Для этого запустим текстовый редактор Блокнот (можно использовать и другой текстовый редактор).

Веб-страница, обычно, как тексты начинается с заголовка. Так как эта наша первая веб-страница, то введем заголовок «Это моя первая веб-страница». Для этого в рабочее поле Блокнота записывается текст как на рисунке 1.

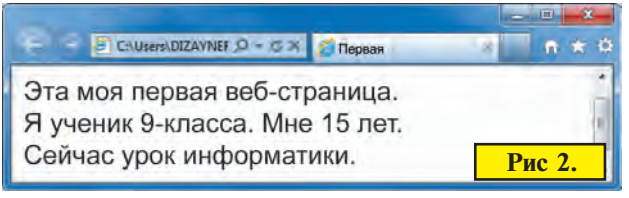
В этом тексте <HTML>, </HTML>, <HEAd>, </head>, <title>, </Title>, <BODY>, </Body> и
 являются тэгами языка HTML, и означают: <HTML> – начало веб-стра-



ницы, <HTML> – конец веб-страницы, <HEAd> – начало раздела заголовка, <HEAd> – конец раздела заголовка, <title> – начало ввода имени, </Title> – конец ввода имени, <BODY> – начало раздела информации, </BODY> – конец раздела информации,
 – вывод продолжения текста с новой строки.

Текст, введенный выше, сохраним под именем **«mypage1.html»** в каталоге, например, в папке «Gulnoza» каталога «Мои документы». Теперь можно увидеть, что знак файла изменился в соответствии с браузером.

Если открыть веб-страницу браузером Internet Explorer, она отражается как показано на рисунке 2:



Сравнивая HTML-документ и веб-страницу, заключаем:

1) имя файла (в примере, **mypage1**) не отражается в веб-странице;

2) название веб-страницы (например, «Первая») отражается в строке заголовка браузера;

3) предложение (в примере, **«Мне 15 лет»**), записанное в тексте веб-страницы без специальной команды, отражается в продолжении предыдущего предложения.

Текст, написанный в HTML, не всегда выдает ожидаемый результат. Приходится несколько раз его редактировать, чтобы привести в нужный вид. Для этого приходится запускать браузер, отыскивать HTML-документ из диска и

загружать, редактировать и сохранять текст, заново запускать браузер, отыскивать отредактированный HTML-документ и запускать. Обычно, чтобы привести веб-страницу в нужный вид, приходится выполнять перечисленные действия несколько раз. Эти действия можно выполнить удобным способом.

Способ 1. Если нужно отредактировать веб-страницу, запущенную браузером Internet Explorer, следует направить указатель мыши на веб-страницу и нажать правую кнопку. Если выбрать из контекстного меню раздел Просмотр HTML-кода, откроется текстовый редактор Блокнот с загруженным HTML-документом веб-страницы. Отредактированный HTML-документ сохраняется. Если выбрать кнопку **«Обновить»** ( ) в панели инструментов Internet Explorer, то в поле информации веб-страница обновляется (отражается отредактированная веб-страница).

Способ 2. В меню **Вид** выбирается раздел Просмотр HTML-кода и редактируется, как при способе 1.

Вопросы и задания

1. *Как называются команды языка HTML?*
2. *Что понимается под HTML-документом? Какими могут быть расширения HTML-документа?*
3. *Расскажите о типах тэгов.*
4. *Каким тэгом начинается HTML-документ?*
5. *Покажите на примере присвоение имени веб-странице.*
6. *Где отражается имя веб-страницы?*
7. *В каких местах HTML-документа можно написать имя веб-страницы?*

Упражнения

1. Подготовьте текст в текстовом редакторе для веб-страницы «Моя родная страна» и сохраните его под именем «Моя родная страна».

2. Переведите текстовый файл «Моя родная страна» в HTML-документ «Моя Родина». Отредактируйте веб-страницу с помощью браузера.

3. Подготовьте простую текстовую веб-страницу с заголовком «Наша школа».

Урок 51. Ввод текста на веб-страницу

В тексте веб-страницы можно освещать разные темы. В таком случае приходится вводить несколько заголовков. Например, на свою личную веб-страницу вы можете ввести

информацию о себе, о своих интересах, о любимых фильмах. Тогда можете выбрать заголовки «О себе», «Мои хобби», «Любимые фильмы». Иногда веб-страница выглядит привлекательней, если подобрать разные размеры шрифта заголовков.

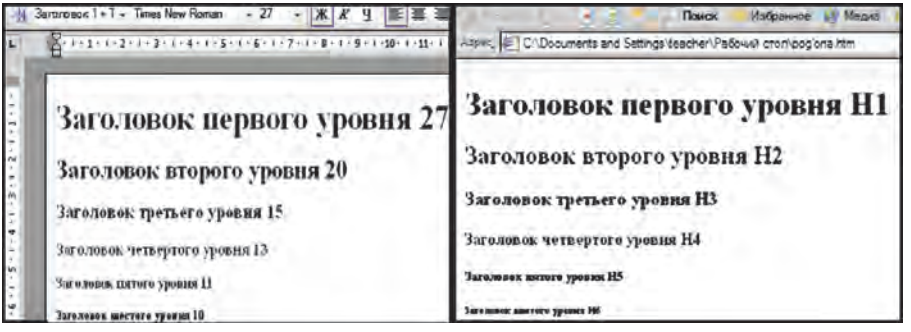
Файл Правка Формат Вид Справка

```
<HTML>
<h1>Заголовок первого уровня    H1</h1>
<h2>Заголовок второго уровня    H2</h2>
<h3>Заголовок третьего уровня   H3</h3>
<h4>Заголовок четвертого уровня H4</h4>
<h5>Заголовок пятого уровня     H5</h5>
<h6>Заголовок шестого уровня    H6</h6>
</html>
```

Язык HTML предоставит возможность вставить заголовки 6 различных уровней. Для этого служат парные тэги **<H1>, <H2>, <H3>, <H4>, <H5>, <H6>** (первая буква "H" от английского «Heading» — заголовок). Следовательно, для каждого из них есть соответствующие закрывающие тэги (</H1>, ..., </H6>).

При работе с программой MS Word, вы познакомились с такими понятиями как размер шрифта (например, 27), тип шрифта (например, Times New Roman), стиль записи (например, Заголовок 1), начертание шрифта (например, полужирный).

На основе этого заголовки документа MS Word и языка HTML можно сравнить следующим образом:



Ввод текста на веб-страницу можно осуществлять как в текстовых редакторах. Текст, введенный в HTML-документ, форматируется браузером относительно размера поля информации. И поэтому текст HTML-документа отражается

на окне браузера несколько иначе. Чтобы текст отражался в таком виде, как мы хотим, следует использовать тэги языка HTML, влияющие на формат шрифта.

Для описания абзацев применяется парный тэг **<P>**. Этот тэг записывается в начале абзаца. Он оставит пустую строку перед текстом, записанным после себя. Как было сказано, непарный тэг **
** применяется для переноса продолжения текста на новую строку. Этот тэг можно применить и в случае, если нужно пропустить строку.

Тэги HTML можно написать с параметрами (атрибутами). Параметры записываются после имени тэга, разделенные пробелом. Если параметрам необходимо присвоить значения, то они записываются после знака равенства в кавычках или без кавычек.

Для установления типа шрифта применяется парный тэг **<FONT>** (шрифт) с параметром **FACE: <FONT FACE= "тип шрифта">** текст **</FONT>**. Вам известно, что к типам шрифта относятся **Times New Roman, Verdana, Elephant.**

Как и в MS Word, на веб-странице можно использовать несколько типов шрифта. Иногда браузер может не поддерживать определенный тип шрифта. Поэтому целесообразно указывать несколько значений параметра FACE:

<FONT FACE = "Times New Roman", "Arial Black", "Elephant">.

Браузер читает значение параметра слева направо и найдет поддерживаемый тип шрифта. Следующий пример показывает применения типа шрифта:

```
<html>
<title>Тип шрифта</title>
<font face="Times New Roman">Тип шрифта -
Times New Roman</font>
<br><font face="Verdana","Arial Narrow">Тип шрифта -
Verdana</font>
<br><font face="Elephant", "Bodoni MT Black">Тип
шрифта - Elephant</font>
</html>
```

Тип шрифта – Times New Roman
Тип шрифта – Verdana
Тип шрифта – Elephant

Иногда, чтобы привлечь внимание к отдельным словам, используют некоторые приемы. В документе MS Word такого эффекта можно было добиться изменением шрифта (полужирный, курсив или подчеркнутый). Следующие тэги языка HTML предоставляют такие возможности:

<B> – полужирный (Bold)	или вместо них	**<Strong>**
<I> – курсив (Italik)		**<Em>** или **<Cite>**
<U> – подчеркнутый (Underline)		–

Применяя тэги, рассмотренные выше, части текста веб-страницы можно привести в виде полужирный и курсив, курсив и подчеркнутый и другие:

```
<HTML><H1>Выделение слов в тексте</H1>
<p>
Чтобы привлечь внимание, выделяют некоторые слова.
<BR>Такого эффекта можно добиться изменением начертания в виде
<B>полужирный</B>,<I> курсив</I> или <U>подчеркнутый</U>.
<BR>Применяя вместе тэги, текст можно привести в виде
<B><I>полужирный и курсив</B>,<U> курсив и подчеркнутый</I>,
<B>полужирный и подчеркнутый</B></U><BR>и другие.
</HTML>
```

Выделение слов в тексте

Чтобы привлечь внимание, выделяют некоторые слова.
Такого эффекта можно добиться изменением начертания в виде **полужирный,** *курсив* или подчеркнутый.
Применяя вместе тэги, текст можно привести в виде ***полужирный и курсив,*** *курсив и подчеркнутый,* **полужирный и подчеркнутый**
и другие.

В программе MS Word с помощью инструментов  можно было выровнять текст на странице (т.е. по левому краю, по центру, по правому краю или по ширине страницы) различными способами.

Язык HTML тоже предоставит возможность выравнивания текста на веб-странице и осуществляется параметром **ALIGN** (англ., выравнивать) парного тэга **<P>** (или парных тэгов **<H1>, <H2>, <H3>, <H4>, <H5>, <H6>):**

<P ALIGN= ″значение параметра выравнивания″>.

Значениями параметра выравнивания служат ″Left″ (слева), ″Right″ (справа), ″Center″ (по центру) и ″Justify″ (по ширине).

Рассмотрим пример расположения абзацев на веб-странице:

```
<html>
<h2>Выравнивание текста на веб-странице</h2>
<p align="left">Эта строка выравнивается по левому краю страницы.
<h5 align="right">Эта строка выравнивается по правому краю страницы.</h5>
<p align="center">Эта строка выравнивается по центру страницы.
<h4 align="justify">Эта строка выравнивается по ширине страницы.
В программе MS Word вы ознакомились со способами выравнивания текста.
На языке HTML эти действия осуществляются по-другому.
</html>
```

Выравнивание текста на веб-странице

Эта строка выравнивается по левому краю страницы.

Эта строка выравнивается по правому краю страницы.

Эта строка выравнивается по центру страницы.

Эта строка выравнивается по ширине страницы. В программе MS Word вы ознакомились со способами выравнивания текста. На языке HTML эти действия осуществляются по-другому.

Введенный текст и его вид в браузере может отличаться в связи с тем, что обычно браузеры игнорируют лишние пробелы (выбрасывают). Иногда требуется, чтобы текст отражался на веб-странице, как введено в HTML-документе.

Например, если нужно вставить на веб-страницу стихи или рисунки, полученные с помощью обычных символов, то форматирование текста нельзя доверить браузеру. В таких случаях используют парный тэг <PRE>. С помощью этого тэга текст выводится так, как написан в HTML-документе. Следующий пример наглядно продемонстрирует работу тэга **<PRE>**.

Между тэгами <PRE> и </PRE> бесполезно использовать тэги <P>,
, браузер просто игнорирует эти тэги. Хотя с помощью тэга <PRE> удобно ввести текст на веб-страницу, лучше его использовать в крайних случаях. Потому что браузер отформатирует текст по размеру информационного поля. Если использован тэг <PRE>, браузер не

```
<html>
<pre>
Весна!      Очень красивый сезон!
  Весна!      Очень красивый сезон!
    Весна!      Очень красивый сезон!
</pre>
<P>
<br>Весна!      Очень красивый сезон!
<br>  Весна!      Очень красивый сезон!
        Весна!      Очень красивый сезон!
</html>
```

форматирует текст и не поместившиеся части текста не будут видны.

Вопросы и задания

1. *Сколько уровней заголовков в HTML-4?*
2. *Чем отличаются уровни заголовков?*
3. *Каким тэгом устанавливается абзац?*
4. *Какой тэг переводит остаток текста на новую строку?*
5. *Приведите примеры записи текста шрифтами полужирный, курсив и подчеркнутый.*
6. *Приведите примеры способов выравнивания текста на веб-странице.*
7. *Какой тэг применяется, чтобы браузер не отформатировал текст?*

Упражнения

1. Создайте веб-страницу, отражающую текст гимна нашей республики. В ней выберите разные типы шрифтов для строк.

2. Создайте веб-страницу, отражающую на различных уровнях заголовков названия республики, области, города, района, улицы, где расположена ваша школа.

3*. Создайте веб-страницу под названием «Наш класс». Используйте в ней различные тэги форматирования.

Урок 52. Повторение темы «Ввод текста на веб-страницу»

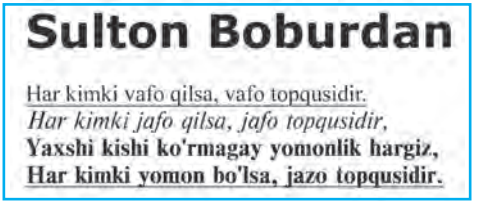
1. Создайте веб-страницу под названием «Моя семья». Используйте в ней различные тэги форматирования. Добавьте в нее простой рисунок (домик, ёлка и т.д.).

2. Создайте веб-страницу, в которой расположите названия предметов на разных уровнях заголовков, по мере вашей заинтересованности.

3*. Сделайте так, чтобы на веб-странице под названием «Наш класс» во всех предложениях подлежащие выделялись полужирным и курсивным шрифтом, а сказуемые – курсивным и подчеркнутым шрифтом.

4*. Создайте веб-страницу под названием «Великие соотечественники». На ней выровненные по центру страницы имена соотечественников должны быть в заголовках 1 уровня, а текст, рассказывающий об их наследии, выровненный справа, должен быть в заголовках 3 уровня.

5*. Создайте веб-страницу под названием «Рубаи» в следующем виде.



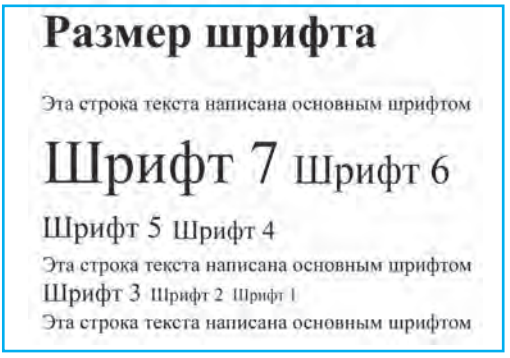
Урок 53. Размер и цвет шрифта, фон веб-страницы

На веб-страницах, созданных на предыдущих уроках, использовались одинаковые шрифты. Размеры шрифтов изменили с помощью тэгов заголовков. На веб-страницах Интернета можно увидеть, что там используются шрифты различных размеров и цветов.

Для установления размера шрифта на языке HTML тэг **<FONT>** применяется с параметром **SIZE** (размер). Этот тэг не меняет размер шрифта в HTML-документе, изменение размера шрифта можно увидеть браузером.

На веб-страницах можно использовать 7 размеров шрифта. Они упорядочены с 1 по 7, 1 – наименьший и 7 – наибольший, размеры. Для установления размера шрифта равного 5 записывается тэг **<FONT SIZE=5>**. Для возврата к основному шрифту добавляется закрывающий тэг **</FONT>**.

```
<html>
<h1>Размер шрифта</h1>
<p>Эта строка текста написана основным шрифтом<br>
<font size=7>Шрифт 7<font size=6> Шрифт 6<br>
<font size=5>Шрифт 5<font size=4> Шрифт 4</font>
</font></font></font>
<br>Эта строка текста написана основным шрифтом<br>
<font size=3>Шрифт 3</font>
<font size=2> Шрифт 2<font size=1> Шрифт 1
</font></font></font>
<br>Эта строка текста написана основным шрифтом
</html>
```



Если вы заметили, сколько раз мы выбирали размер шрифта, столько же раз и закрывали соответствующий ТЭГ.

Использование на веб-страницах различных цветов придает им более красивый и привлекательный вид. На веб-страницах наряду с цветом шрифта (символа) и текста, можно изменять цвет фона. Для установления цвета шрифта применяется параметр **COLOR** тэга **<FONT>**:

<FONT COLOR= "# код цвета">

Обратите внимание, что коду цвета обязательно должен предшествовать значок «решетка» – #.

Белый	#FFFFFF	White
Черный	#000000	Black
Красный	#FF0000	Red
Зеленый	#00FF00	Green
Синий	#0000FF	Blue
Желтый	#FFFF00	Yellow
Розовый	#FF00FF	Magenta

Код цвета основан на системе RGB (Red-красный, Green-зеленый, Blue-синий). Нужный цвет получается комбинированием в разных пропорциях этих цветов. Каждый из цветов отдельно кодируется числами в шестнадцатеричной системе счисления от 00 по FF (256 штук). Для основных цветов вместо кода можно использовать названия на английском языке. В таблице приведены коды и названия некоторых цветов.

Выбор цвета можно увидеть в следующем примере:

```
<html>
<font color="red"> Все</font>
<font color="blue"> слова</font>
<font color="magenta"> этой</font>
<font color="green"> веб-страницы</font>
<font color="yellow"> в разных</font>
<font color="black"> цветах</font>
</html>
```

Все слова этой веб-страницы в разных цветах

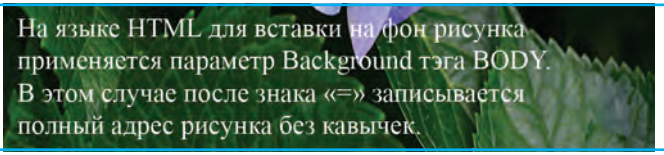
Изменение цвета текста и фона веб-страницы осуществляется с помощью парного тэга **<BODY>** с параметрами, соответственно, **Text** (текст) и **Bgcolor** (background color, т.е. цвет фона). После этих параметров ставится знак "=" и в кавычках записывается значок "#" и код цвета или название цвета. Следует отметить, что этот тэг с параметром Text не меняет цвет шрифта, измененный тэгом <FONT>. Следующий пример показывает применение тэгов цвета:

Все {Эта часть текста отражается розовым цветом, соответственно цвету текста} слова этой веб-страницы {Эта часть текста отражается розовым цветом, соответственно цвету текста} в разных цветах {Эта часть текста отражается розовым цветом, соответственно цвету текста}

```
<html><body bgcolor="#ffddd0" text="#ff00ff"><font
color="red"> Все </font>{Эта часть текста отражается
розовым цветом, соответственно цвету текста}
<font color="blue"> слова </font>
<font color="magenta"> этой</font>
<font color="green"> веб-страницы  </font>
{Эта часть текста отражается розовым цветом,
соответственно цвету текста}
<font color="yellow"> в разных</font>
<font color="black"> цветах</font>
{Эта часть текста отражается розовым цветом,
соответственно цвету текста}
</body></html>
```

В программе MS Word можно было изменить цвет фона или вставить в фон рисунок. На языке HTML для вставки на фон рисунка применяется параметр **Background** тэга **<BODY>**. В этом случае после знака "=" записывается полный адрес рисунка без кавычек. Если рисунок находится в каталоге вместе с HTML-документом, то записывается только имя файла (например, Gul.jpg).

```
<html>
<body background=Nastarin.jpg>
<body text="#ffffff">
<p>На языке HTML для вставки на фон рисунка
<br>применяется параметр Background тэга BODY.
<br>В этом случае после знака «=» записывается
<br>полный адрес рисунка без кавычек.
</body></html>
```



Для быстрой загрузки веб-страницы расширение рисунка обычно соответствует формату JPEG и GIF.

Но рисунки формата BMP также можно использовать. Об этом расскажем на следующем уроке.

Вопросы и задания

1. *Сколько разных цветов можно использовать на языке HTML?*
2. *При помощи какого ТЭГа установливается размер шрифта на Web-странице?*
3. *При помощи какого тэга устанавливается цвет шрифта на Web-странице?*
4. *Каким тэгом устанавливается цвет текста?*
5. *Расскажите о работе параметров тэга <BODY>.*
6. *Покажите на примере выбор цвета для фона на веб-странице.*
7. *Расскажите о вставке рисунка на фон веб-страницы.*

Упражнения

1. На веб-странице «Великие соотечественники» для разных слов выберите разные цвета.

2. На веб-странице «Моя семья» поменяйте цвет слов, текста и фона.

3. На фон веб-страницы «Наш класс» вставьте рисунок.

Урок 54. Повторение темы «Размер и цвет шрифта, фон веб-страницы»

1. Напишите Html-код, чтобы на веб-странице ваша фамилия, имя и отчество были выполнены шрифтами различных размеров.

2. Напишите Html-код, чтобы на веб-странице ваш район, школа, класс, фамилия и имя были выполнены шрифтами разных размеров и цветов.

3. Сначала в Html-коде напишите тэг для заливки фона, после него тэг для вставки рисунка на фон. Объясните вид полученной веб-страницы.

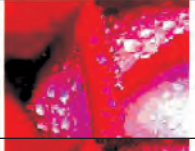
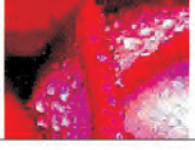
4. Создайте веб-страницу под именем «Исторические города нашей Родины». Отформатируйте на ней текст так, чтобы названия городов различались цветом и размером шрифта. Информацию о городах поместить с абзаца и выравнить по центру.

5. Создайте веб-страницу под названием «Моя любимая профессия». Для вставки в фон рисунка скопируйте его из коллекции программы MS Word (подсказка: сначала вставить нужный рисунок в документ MS Word, потом копировать в MS Paint и нужную часть рисунка сохранить в файл).

Урок 55. Графика на веб-странице

Самый эффективный метод «оживления» веб-страницы — вставка различных рисунков. Веб-страница, состоящая только из текста, если даже содержит богатую информацию, может оказаться скучной. С другой стороны, вставка на веб-страницу чересчур много рисунков увеличит ее объем. А загрузка файла большого объема из сети Интернета потребует много времени. И поэтому на веб-страницу целесообразно вставлять рисунки малого объема. В сети Интернет используются, в основном, рисунки формата **jpeg** и **gif**. Потому что, во-первых, уменьшится, объем веб-страницы, во-вторых, браузер сможет обработать такие рисунки без дополнительных программ. На самом деле, если файл рисунка формата **bmp** перевести в файл формата **jpeg**, то объем файла рисунка сократится в несколько раз.

Для перевода файла рисунка с одного формата в другой используются специальные программы (ACDSee, Photoshop и др.). Для сравнения приведем следующий пример:

Рисунки	Формат	Размер рисунка	Объем графического файла
	BMP	130 x 100 пикселей	38,3 килобайта
	JPEG	130 x 100 пикселей	4,44 килобайта
	GIF	130 x 100 пикселей	6,5 килобайта

Для вставки рисунка на веб-страницу применяется непарный тэг **< IMG >** (image — изображение, рисунок). Для указания имени файла, содержащего рисунок, добавляется параметр **SRC** (source – источник). Например, если имя нужного файла «mypic.jpg», то в HTML-документ добавляется следующая строка:

<IMG SRC = ″mypic.jpg″>,

где **mypic.jpg** значение параметра SRC (кавычки не обязательны).

На предыдущих уроках вы ознакомились с параметром ALIGN для выравнивания текста на веб-странице. Параметр ALIGN может служить для вставки рисунка на левую или на правую стороны веб-страницы. Например, запись <IMG ALIGN=″right″ SRC=″mypic.jpg″> или <IMG SRC=″mypic.jpg″ ALIGN=″right″> вставить рисунок ″mypic.jpg″ на правую сторону веб-страницы.

Этот параметр применяется для размещения текста в различных положениях вокруг рисунка. На веб-страницах текст и рисунок можно размещать в зависимости от значения параметра ALIGN в следующих положениях:

MIDDLE	Середина рисунка выравнивается по низу текущей (за рисунком) строки
ABSMIDDLE	Середина рисунка выравнивается по середине текущей строки
BOTTOM	Нижняя граница рисунка выравнивается по низу текущей строки
TOP	Верхняя граница рисунка выравнивается по самому большому символу текущей строки
LEFT	Если рисунок помещен на левом поле, текст записывается справа от рисунка
RIGHT	Если рисунок помещен на правом поле, текст записывается слева от рисунка

Например: 1. Запись <IMG ALIGN=″Top″ SRC= =″kub.bmp″> на веб-странице выравнивает рисунок ″kub.bmp″ по самому большому символу текущей строки:

```
<html>
Рисунок куба<br>
<img src="kub.bmp" align="top"> У куба 6 боковых сторон,
12 ребер, 8 вершин.
</html>
```

Рисунок куба

У куба 6 боковых сторон, 12 ребер, 8 вершин.

2. При записи <IMG SRC=″kub.bmp″ ALIGN=″right″> рисунок ″kub.bmp″ помещается на веб-странице на ее правом поле, верхняя граница рисунка выравнивается по самому большому символу текущей строки, и текущая строка записывается слева от рисунка:

```
<html>
Рисунок куба<br>
<img src="kub.bmp" align="right"> У куба 6 боковых сторон,
12 ребер, 8 вершин.
</html>
```

Рисунок куба
У куба 6 боковых сторон, 12 ребер, 8 вершин.

Следует отметить, что расположение текста и рисунка на веб-странице зависит от тэгов, применяемых для форматирования текста. Чтобы увидеть это, достаточно убрать тэг
 во втором примере.

При вставке рисунка на веб-страницу можно установить его размеры.

Для этого применяют параметры **WIDTH** (ширина) и **HEIGHT** (высота), значения которых задаются в пикселях или в процентном соотношении относительно истинного размера (но некоторые браузеры не поддерживают проценты). Например, запись

<IMG SRC=″mypic.jpg″ WIDTH=50 HEIGHT=100>

позволяет вставить рисунок mypic.jpg независимо от его истинных размеров (рисунок из таблицы выше, размер 130x100 пикселей) на веб-страницу с размером 50x100 пикселей.

Не забудьте: увеличение размеров рисунка может привести к потере качества! Целесообразно привести рисунок в нужный размер с помощью графических редакторов и только потом вставить на веб-страницу!

```
<html>
<p align= "justify">
Чтобы вокруг рисунка оставить (отступ от рисунка) пустое
пространство (space), применяются параметры HSPACE и
VSPACE, значения которых задаются в пикселях.<br>
<img src="lola.jpg" align=left width=50 height=100> HSPACE
- оставляет пустое пространство слева и справа от<br>
рисунка. <br>VSPACE - оставляет пустое пространство сверху
и снизу от<br>рисунка.
</html>
```

Чтобы вокруг рисунка оставить (отступ от рисунка) пустое пространство (space), применяются параметры HSPACE и VSPACE, значения которых задаются в пикселях.

HSPACE – оставляет пустое пространство слева и справа от рисунка.
VSPACE – оставляет пустое пространство сверху и снизу от рисунка.

При вставке рисунка на веб-страницу вокруг рисунка может не оказаться пустого пространства, т.е. текст или другой рисунок может «касаться» рисунка (пример 1, выше). Чтобы вокруг рисунка оставить (отступ от рисунка) пустое пространство (space), применяются параметры **HSPACE** (оставляют пустое пространство слева и справа от рисунка) и **VSPACE** (оставляют пустое пространство сверху и снизу от рисунка), значения которых задаются в пикселях.

Например, запись

<IMG SRC="lola.jpg" ALIGN=left HSPACE=15 VSPACE=15>

вставит на веб-странице, оставив слева, справа, сверху и снизу оставив пустое пространство (в виде рамки) размером 15 пикселей рисунок «lola.jpg»:

```
<html>
<p align= "justify">
Чтобы вокруг рисунка оставить (отступ от рисунка) пустое
пространство (space), применяются параметры HSPACE и
VSPACE, значения которых задаются в пикселях.<br>
<img src="lola.jpg" align=left hspace=15 vspace=15> HSPACE
- оставляет пустое пространство слева и <br> справа от
рисунка. <br>VSPACE - оставляет пустое пространство сверху
и <br> снизу от рисунка.</html>
```

Чтобы вокруг рисунка оставить (отступ от рисунка) пустое пространство (space), применяются параметры HSPACE и VSPACE, значения которых задаются в пикселях.

HSPACE – оставляет пустое пространство слева и справа от рисунка.
VSPACE – оставляет пустое пространство сверху и снизу от рисунка.

Чтобы вокруг рисунка создать (черную) рамку, применяют параметр **BORDER** (граница).

Значение этого параметра тоже измеряется в пикселях, например:

<IMG SRC="lola.jpg" BORDER=9 >.

Попробуйте создать самостоятельно веб-страницу, содержащую рисунок с рамкой.

Вопросы и задания

1. *Какие графические форматы вы знаете?*
2. *С помощью программы PAINT переведите рисунок с формата BMP в форматы JPEG и GIF.*
3. *Как помещают рисунок на веб-страницу?*
4. *Какие параметры тэга <IMG> вы знаете?*
5. *Как организуется вставка рисунка на веб-страницу с левой или с правой стороны?*
6. *Как можно изменить размеры рисунка при вставке на веб-страницу?*
7. *Какие параметры тэга <IMG> оставляют вокруг рамки пустое пространство*?

Упражнения

1. На веб-страницу под названием «Великие соотечественники» вставьте разные рисунки, соответствующие той области, в которой они работали.

2. На веб-страницу под именем «Моя любимая профессия» вставьте рисунки, относящиеся к профессиям. Вокруг рисунка создайте рамку размером 15 пикселей.

3. На веб-страницу под именем «Моя семья» вставьте рисунки, относящиеся к профессиям родственников. Используйте разные способы размещения.

Уроки 56—57. Повторение темы «Графика на веб-странице»

1. На веб-страницу «Наш класс» вставьте 3 рисунка.
2. На веб-страницу «Исторические города нашей Родины» вставьте рисунки так, чтобы размеры и способы их размещения отличались (подсказка: рисунки можно импортировать из коллекции ...).
3. Создайте веб-страницу под названием «Домашние животные». Вставьте на нее рисунки животных так, чтобы возле рисунков размещались названия животных (подсказка: рисунки можно импортировать из коллекции MS Word).
4. Создайте веб-страницу под названием «Устройство компьютера». С помощью графического редактора создайте нужные рисунки для этой веб-страницы.
5. Создайте веб-страницу под названием «Мои друзья». Оформите ее рисунками и сопроводите информацией об их любимых профессиях.

Урок 58. Вставка списка на веб-страницу

В документе текстового процессора MS Word создавали два вида списка: маркированные (с помощью кнопки ) и упорядоченные (с помощью кнопки ), например:

Маркированный список	**Маркированный список**	**Маркированный список**	**Упорядоченный список**	**Упорядоченный список**
• Информатика • Математика • История	o Информатика o Математика o История	♦ Информатика ♦ Математика ♦ История	1. Информатика 2. Математика 3. История	A. Информатика B. Математика C. История

Для ввода на веб-страницу используются парные тэги <UL> (unordered list – неупорядоченный, т.е. маркированный список) или <OL> (ordered list – упорядоченный список). Обычно, в HTML-документе данные, расположенные после тэгов <OL> и <UL>, отражаются браузером отступом вправо. Для выделения элементов списка применяется непарный тэг <LI> (list item – элемент списка). Элемент списка, начинающийся с тэга <LI>, всегда размещается с новой строки. Например, первый столбец приведенной выше таблицы, отражающий маркированный список, в HTML-документе записывается следующим образом:

```
<html><UL>
<LI>Информатика<LI> Математика<LI> История
</UL></html>
```

Естественен вопрос: как можно изменить вид маркера?

Тэг <UL>, вводящий маркированный список, можно использовать с параметром **TYPE**. Значениями параметра являются **disk** (круг), **circle** (окружность), **square** (окрашенный квадрат), например <UL TYPE = square>.

Если тэг <UL> записывается без параметра, то по умолчанию браузер маркированный список отражает с маркером disk. Тэг <LI> тоже можно применять с параметром:

```
<html><ul>
<li type=disk> Информатика
<li type=circle> Математика
<li type=square> История
</ul></html>
```

- Информатика
- Математика
- История

Если нужно отражать упорядоченный список таблицы, то фрагмент HTML-документа и вид в браузере выглядят так:

```
<html><ol>
<li> Информатика<li> Математика<li> История
</ol></html>
```

1. Информатика
2. Математика
3. История

Если порядковый номер списка должен начинаться с числа отличного от 1 (например, с 3), то тэг <OL> используется вместе с параметром **START**, например: <OL start = 3>. Если список нужно упорядочить латинскими буквами или римскими цифрами, то тэг <OL> используется вместе с параметром TYPE. Тэг <LI> также может применяться с параметрами TYPE и VALUE.

В следующем примере можно наглядно увидеть упорядоченные списки разного вида.

Иногда встречаются списки, в которых вместо маркера используются графические изображения, а это выглядит красиво и привлекательно. В этом случае следует учесть, что объем файла увеличится в несколько раз. Так как это список пользователя, тэг <LI> не применяется. Список орга-

```
<html>
<ol start=3><li> Информатика<li> Математика<br>...
<li value=17> История</ol>
<ol type=A><li> Информатика<li> Математика</ol>
<ol type=a><li> Информатика<li> Математика</ol>
<ol type=I start=5><li> Информатика <li> Математика</ol>
<ol type=i><li> Информатика<li type=A> Математика</ol>
</html>
```

низуется с помощью тэгов <P> и
, как в следующем случае:

```
<html><ul>
<h2>Деревья</h2><br>
<h3><img src="Barglar.gif > Ёлка <br>
<img src="Barglar.gif> Ива<br>
<img src="Barglar.gif"> Черешня</h3>
</ul></html>
```

Деревья

- Ёлка
- Ива
- Черешня

Запомните – в этом случае графический файл «Barglar.gif» содержится в одном каталоге с веб-страницей.

В сети Интернет очень много обучающих веб-сайтов, с помощью которых можно изучить некоторый предмет или его часть или использование некоторого программного средства. Известно, что некоторый предмет или программное средство включает необходимые понятия и определения.

Чтобы описать эти понятия и определения на веб-страницах, используется парный тэг **<DL>** (definition list – список определений). Внутри этого тэга используются непарные тэги **<DT>** (definition term – описываемое понятие) и **<DD>** (definition description – описание).

Следующий пример показывает функции этих тэгов:

```
<html><dl><dt>Информатика
<dd>это основанная на использовании компьютерной техники
дисциплина, изучающая структуру и общие свойства
информации, а также закономерности и методы её создания,
хранения, поиска, преобразования, передачи и применения в
различных сферах человеческой деятельности.
<dt>Дескриптор или тэг<dd> это команды языка HTML, они
записываются между знаками «<» и «>», tag в переводе с
английского – метка, ярлык, признак.
</dl></html>
```

Информатика
это основанная на использовании компьютерной техники дисциплина, изучающая структуру и общие свойства информации, а также закономерности и методы её создания, хранения, поиска, преобразования, передачи и применения в различных сферах человеческой деятельности.

Дескриптор или тэг
это команды языка HTML, они записываются между знаками «<» и «>», tag в переводе с английского – метка, ярлык, признак.

Встречаются списки, внутри которых вводятся другие списки. В этом случае получаются вложенные или многоуровневые списки.

В следующем примере с помощью известных вам тэгов показан многоуровневый список.

```
<html><ul>
<b>Спутники некоторых планет</b>
<li> Земля
<ol><li> Луна </ol>
<li> Марс
<ol><li> Фобос <li> Деймос </ol>
</ul></html>
```

Спутники некоторых планет
- Земля
  1. Луна
- Марс
  1. Фобос
  2. Деймос

Как видите, составить такого рода списки не представляет трудности, но чтобы не допустить ошибку, надо быть внимательным.

Вопросы и задания

1. *Какие виды списка можно создать в документе MS Word?*
2. *Сколько видов списка можно использовать на веб-странице? Приведите примеры.*
3. *Как организуется упорядоченный список? Приведите примеры.*
4. *Как организуется маркированный список? Приведите примеры.*
5. *Объясните процесс организации списка с графическим изображением.*
6. *Как организуется графически маркированный список?*
7. *Расскажите о списке определений.*

Упражнения

1. Составьте список с римскими цифрами.
2. Составьте список, который начинается с 9.
3. Составьте многоуровневый список.

Урок 59. Вставка таблиц на веб-страницу

Для ввода на веб-страницу таблиц используются парные тэги <**TABLE**> (table – таблица), <**TR**> (table row – строка таблицы), <**TH**> (table header – заголовок внутри таблицы) и <**TD**> (table data – данные таблицы). Тэг <TABLE> указывает на начало таблицы, тэг </TABLE> – конец таблицы, тэг <TR> – строку таблицы, тэг <TD> – столбец таблицы. Тэг <TH> означает ячейку с заголовком, в которой данные отражаются в полужирном шрифте и выровненные от середины ячейки (т.е. ALIGN=Center и VALIGN=Middle, здесь V – вертикальное направление). Тэг <TH> не используется без тэга <TR>. Если на веб-странице должна отражаться граница таблицы, то применяется параметр BORDER. Значение параметра BORDER задается в пикселях и воздействует только на толщину внешних границ таблицы.

В HTML-документе таблица состоит из строк. Каждая строка делится на столбцы. Тэги <TR> и <TD> в свою очередь закрываются тэгами </TR> и </TD>.

Например, возьмем отрывок HTML-документа, отображающий таблицу:

Название месяца	**Сезон**	**Порядок месяца**
Январь	2-месяц зимы	Первый месяц года
Декабрь	1-месяц зимы	Последний месяц года

Он записывается в следующем виде:

```
<TABLE>
<TABLE BORDER>
<TR><TH>Название месяца </TH><TH> Сезон </TH><TH> Порядок месяца </TH></TR>
<TR><TD>Январь</TD><TD>2 месяц зимы</TD> <TD>Первый месяц года</TD> </TR>
<TR><TD>Декабрь</TD><TD>1 месяц зимы</TD> <TD> Последний месяц года</TD> </TR>
</TABLE>
```

Следующий пример отражает таблицу с границей и без границы:

```
<HTML><TABLE>
<TR><TH>Название месяца</TH><TH>Сезон</TH><TH>Порядок
месяца
</TH></TR><TR><TD>Январь</TD><TD>2 месяц зимы</TD>
<TD>Первый месяц года</TD></TR>
<TR><TD>Декабрь</TD><TD>1 месяц зимы</TD>
<TD>Последний месяц года</TD></TR></TABLE>
<TABLE><TABLE BORDER=7>
<TR><TH>Название<br>месяца</TH><TH>Сезон</TH>
<TH>Порядок месяца</TH></TR><TR><TD>Январь</TD>
<TD>2 месяц зимы</TD><TD>Первый месяц года</TD></TR>
<TR><TD>Декабрь</TD><TD>1 месяц зимы</TD >
<TD>Последний месяц<br>года</TD></TR>
</TABLE></HTML>
```

Название месяца	Сезон	Порядок месяца
Январь	2 месяц зимы	Первый месяц года
Декабрь	1 месяц зимы	Последний месяц года

Название месяца	Сезон	Порядок месяца
Январь	2 месяц зимы	Первый месяц года
Декабрь	1 месяц зимы	Последний месяц года

Если необходимо ввести заголовок в таблицу, то применяется парный тэг <**CAPTION**> (заголовок) и нужно написать перед первым тэгом <TR>. Функции и значения параметров ALIGN и VALIGN этого тэга приведены в следующей таблице:

ALIGN	VALIGN	Функции
TOP	Не вводится	Заголовок помещен выше таблицы и выравнивается по середине таблицы
BOTTOM	Не вводится	Заголовок помещен ниже таблицы и выравнивается по середине таблицы
LEFT	TOP	Заголовок помещен наверху таблицы и выравнивается по левой границе таблицы
LEFT	BOTTOM	Заголовок помещен внизу таблицы и выравнивается по левой границе таблицы
CENTER	TOP	Заголовок помещен наверху таблицы и выравнивается по середине таблицы

CENTER	BOTTOM	Заголовок помещен внизу таблицы и выравнивается по середине таблицы
RIGHT	TOP	Заголовок помещен наверху таблицы и выравнивается по правой границе таблицы
RIGHT	BOTTOM	Заголовок помещен внизу таблицы и выравнивается по правой границе таблицы

В следующем примере можно увидеть возможности параметров ALIGN и VALIGN:

```
<html><table><table border=7>
<caption align=right valign=bottom>ТАБЛИЦА</caption>
<tr><th>Название<br>месяца</th><th>Сезон</th><th>Порядок
месяца </th></tr>
<tr><td>Январь</td><td>2 месяц зимы</td><td>Первый месяц
года</td></tr> <tr><td>Декабрь</td><td>1 месяц зимы
</td><td>Последний месяц<br>года</td></tr>
</table></html>
```

Название месяца	Сезон	Порядок месяца
Январь	2 месяц зимы	Первый месяц года
Декабрь	1 месяц зимы	Последний месяц года

ТАБЛИЦА

Толщина внутренних линий (расстояние между параллельными линиями, разделяющими две ячейки) определяется параметром **CELLSPACING** в пикселях (например, CELLSPACING =5). Расстояние между данными в ячейке и границей ячейки определяется параметром **CELLPADDING** в пикселях (например, CELLPADDING=9).

Ранее мы рассматривали вопрос о расположении рисунка и текста. Вокруг таблицы тоже есть возможность размещения текста, но только слева или справа от таблицы. Для этого тэг <TABLE> используется параметром ALIGN, например: <TABLE ALIGN=LEFT>. Если таблица выравнивается на веб-странице слева, то текст автоматически размещается справа от таблицы и наоборот.

Следует отметить, что для форматирования данных в ячейках можно использовать все тэги форматирования заголовка, текста и рисунка. Для изменения цвета граничных

линий таблиц, строк и столбцов используется параметр BORDERCOLOR.

Используя информацию, приведенную выше, можно создать следующие HTML-документ и веб-страницу:

```
<html>
<table><table align = right Border=7 bordercolor=RED
CELLSPACING=5 CELLPADDING=3>
<TR bordercolor=BLUE> <TH>Название<BR>месяца</TH><TH>Сезон
</TH> <TH>Порядок месяца</TH></TR>
<TR><TD>Январь</TD><TD bordercolor=MAGENTA>2 месяц зимы
</TD><TD>Первый месяц<BR>года</TD>
<TR bordercolor=black><TD>Декабрь</TD><TD>1 месяц зимы
</TD><TD bordercolor=green>Последний месяц года</TD></TR>
</table>
<BR>Следует отметить, что для форматирования данных в
ячейках можно использовать все тэги форматирования
заголовка, текста и рисунка; для изменения цвета граничных
линий таблиц, строк и столбцов используется параметр
BORDERCOLOR.
</html>
```

Следует отметить, что для форматирования данных в ячейках можно использовать все тэги форматирования заголовка, текста и рисунка; для изменения цвета граничных линий таблиц, строк и столбцов используется параметр BORDERCOLOR.

Название месяца	Сезон	Порядок месяца
Январь	2 месяц зимы	Первый месяц года
Декабрь	1 месяц зимы	Последний месяц года

В таблице программы MS Word можно было объединить и разделить ячейки, например:

1 ячейка	Ячейка с двумя объединенными столбцами	
Ячейка с двумя объединенными строками	4 ячейка	Ячейка с двумя объединенными строками
	6 ячейка	

Язык HTML тоже предоставит эти возможности. Для этого тэги <TH> или <TD> используются с параметрами **COLSPAN** (column spanning – объединение столбцов) и **ROWSPAN** (row spanning – объединение строк). Если рассуждать логически, таблицу с разделенной ячейкой можно

получить объединением ячеек другой таблицы. HTML-код таблицы выше выглядит так:

```
<HTML><TABLE><TABLE BORDER CELLPADDING=2><TR>
<TD align=middle><font color=red>1 ячейка</font></TD>
<TD COLSPAN=2 bgcolor=#00ffD0><B><font color=white>Ячейка
с двумя объединенными столбцами</font></B></TD>
<TR><TD ROWSPAN=2 bgcolor=yelIow><B><I><U>Ячейка с двумя
объединенными строками</U></I></B></TD>
<TD bg color=magenta><U>4 ячейка</U></TD>
<TD ROWSPAN=2 bgcolor=black><B><font color=white>Ячейка с
двумя объединенными строками</font></B></TD></TR><TR><TD
bgcolor=#808080><I>6 ячейка</I></TD>
</TABLE></HTML>
```

Над таблицами можно выполнить еще много операций, рекомендуем изучить их самостоятельно.

Вопросы и задания

1. *Как организуется таблица на веб-странице?*
2. *Как форматируются граничные линии таблицы?*
3. *Расскажите о заголовках внутри таблицы.*
4. *Расскажите о заголовке таблицы.*
5. *Как располагаются таблица и текст на веб-странице?*
6. *Приведите примеры форматирования данных таблицы.*
7. *Приведите примеры форматирования граничных линий таблицы.*

Упражнения

1. Создайте веб-страницу для описания тэгов, относящихся к таблицам.
2. На веб-страницу «Наш класс» вставьте таблицу.
3. На веб-страницу «Моя семья» вставьте таблицу, содержащую данных о родственниках.

Урок 60. Повторение тем «Вставка списка на веб-страницу» и «Вставка таблиц на веб-страницу»

1. Создайте многоуровневый список с вложенным маркированным и упорядоченным списком, связанным со школой.
2. На веб-странице «Устройство компьютера» вставьте таблицу с маркированным списком составных частей устройства.
3. Создайте веб-страницу под названием «Мой любимый литературный герой». Название произведения на ней должно отражаться в виде заголовка, а информация об авторе и характеристика героя – в виде таблицы.

4. На веб-страницу под названием «Наш класс» вставьте таблицу с фамилиями и порядковыми номерами из классного журнала 5 отличников вашего класса.

5. На веб-страницу под названием «Моя любимая профессия» введите таблицу с информацией в виде списка об отраслях, связанных с любимой профессией.

Урок 61. Переход (гиперссылка) на веб-странице

Информация, находящаяся на веб-странице, может состоять из нескольких разделов. Возможность быстрого «перехода» с одного раздела на другой ускорит просмотр веб-страницы. Такие переходы на веб-странице составят гиперссылку. Технология гиперссылок в свое время способствовала быстрому и широкому распространению службы WWW.

Гиперссылка, т.е. переход с одного места документа на другое место или на другой документ, осуществляется парным тэгом **<A>**. Тэг имеет параметр **HREF**, значением которого служит адрес места перехода. Место на веб-странице, в котором записан этот тэг, называется **точкой перехода**. Запись тэга <A> с параметром HREF имеет следующий вид:

<A HREF = "#адрес"> текст </A>,

здесь текст — произвольный текст, который выделяет браузер, «адрес» — адрес места (раздел) перехода. Обычно текст, указывающий точку перехода, на экране выделяют синим цветом и подчеркиванием. Адрес тоже может быть произвольным текстом.

В место перехода на веб-странице «адрес» вводится параметром **NAME** тэга <A>. Он должен быть идентичным с «адресом» точки перехода. Общий вид следующий:

<A NAME = "адрес"> текст </A>,

здесь текст – произвольный текст. Обычно, в качестве текста записывается название раздела, начинающегося с этого места веб-страницы. Как видите, «адрес» в <A NAME> отличается от «адреса» в <A HREF> знаком «#».

Чтобы выделить точку перехода на веб-странице, его можно включить в состав списка. Гиперссылка на веб-странице осуществляется нажатием на левую кнопку, указывая указателем мыши на точку перехода. В этом случае цвет текста в точке перехода изменится (обычно в розовый цвет).

```
<HTML>
<H2 ALIGN="center">Гиперссылка на веб-странице</H2>
<UL>
<LI><A HREF="#1-glava">Глава I</A>
<LI><A HREF="#2-glava">Глава II</A>
<LI><A HREF="#3-glava">Глава III</A>
</UL>
<P><H2><A NAME="1-glava">Глава I</A></H2>
<P>Здесь содержится информация относящийся к Главе I
<P><H2><A NAME="2-glava">Глава II</A></H2>
<P>Здесь содержится информация относящийся к Главе II
<P><H2><A NAME="3-glava">Глава III</A></H2>
<P>Здесь содержится информация относящийся к Главе III
</HTML>
```

В приведенном примере точки перехода включены в состав списка. Если на экране видны точки перехода и места перехода, то выполнение перехода будет незаметно.

В точке перехода вместо текста можно вставить рисунок. Для этого в точке перехода вместо текста используется тэг <IMG>. Например, если ввести в HTML-документ запись

<A HREF="#Глава 1"><IMG SRC="mypic.jpg"></A>,

то на веб-странице через рисунок «mypic.jpg» можно перейти в раздел «Глава 1».

С помощью тэга <A> можно организовать переход не только внутри веб-страницы, но и с одной веб-страницы на другую веб-страницу. В этом случае вместо адреса достаточно написать адрес веб-страницы в сети Интернета, т.е. URL-адрес. Например:

<A HREF="http//:www.rambler.ru">Переход в Rambler.ru </A>.

Вместо адреса можно указать имя файла веб-страницы в диске. В этом случае при загрузке откроется веб-страница с диска. Это свойство предоставляет возможность создать **составные веб-страницы (веб-сайты)**.

Составные веб-страницы — совокупность веб-страниц, посвященных одной теме и взаимосвязанных, с возможностью перехода между собой.

Разделение одной темы на части и создание для каждой части отдельной веб-страницы может показаться лишней работой. Но это имеет свои преимущества:

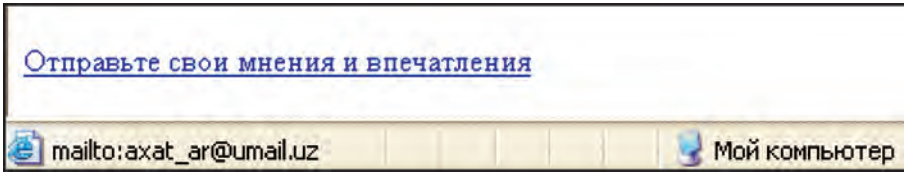
- чем меньше объем информации на веб-странице, тем легче ее редактировать;
- чем меньше объем информации на веб-странице, тем удобней ее прочесть (просмотреть);
- чем меньше объем информации на веб-странице, тем быстрее можно ее «скачать» из сети Интернета.

С помощью тэга <A> можно организовать отправку информации в электронную почту.

Для этого перед адресом электронной почты записывается **mailto** (к почте), например:

<A HREF="mailto:axat_ar@umail.uz>Отправьте свои мнения и впечатления</A>.

Эта гиперссылка с виду особо не отличается от других. Если направить указатель мышки на гиперссылку, в строке состояния отражается адрес электронной почты:



Когда обращаются к этому адресу, браузер открывает окно работы с электронной почтой (если подключена сеть Интернета). Интерфейс этого окна отличается в разных браузерах, но большинство из программ выдают информацию об адресе и имени пользователя и напоминают заполнить строку «Subjekt».

Вопросы и задания

1. *Что вы понимаете под URL-адресом?*
2. *Как организуется гиперссылка на веб-странице?*
3. *Расскажите о точке перехода.*
4. *Что вы знаете о составных веб-страницах?*
5. *Как организуется гиперссылка к электронной почте?*

Упражнения

1. На веб-странице под названием «Моя семья» организуйте переход к информации о членах семьи через их список.
2. На веб-странице «Домашние животные» организуйте переход к информации о животных через их рисунки.
3. Создайте главную веб-страницу под названием «Мои веб-страницы» и организуйте переход к веб-страницам, созданным вами, и обратно на главную веб-страницу.

Урок 62. Формы

Формы на веб-страницах применяются для проведения опросов, диалога между сервером и пользователем или для отбора нужного документа среди представленных документов, например, определение рейтинга веб-страницы, сбор информации о продукте предприятия, знакомства через Интернет. Вопросы в форме отличаются в зависимости от цели формы. Но организация на веб-странице вопросников осуществляется с помощью одинаковых тэгов. На веб-странице эти тэги вместе с текстом вопроса создают окно для ответа. Текст вопроса вводится на веб-страницу в процессе создания.

Окно для ответов по структуре разделяется на два типа:

1. Окно, рассчитанное для ввода произвольного ответа.
2. Окно, рассчитанное для выбора из представленных ответов.

Вопросы можно вводить в любом месте веб-страницы, т.е. структуру формы можно организовать в интересующем вас виде. Диалог можно организовать по-разному. Например, как рассмотрено ранее, вводя на веб-страницу нужные вопросы (как обычный текст) и адрес, можно попросить переслать ответы по электронной почте. Но в этом случае мало кто ответит. Потому что не каждый человек присылает письмо незнакомому человеку. С другой стороны ответы на вопросы формы не требуют особенных усилий для ответа. Для организации формы используется парный тэг **<FORM>**, который имеет параметры **ACTION** и **METHOD.** Параметр ACTION обязательный, так как его значением является URL-адрес. Передачу формы можно осуществлять несколькими методами. Для установления метода передачи используется параметр METHOD. Во многих случаях для передачи формы используется электронная почта.

В этом случае тэгу <FORM> добавляют параметры и их значения METHOD=POST и ACTION= ″mailto: адрес электронной почты″, например:

<FORM METHOD=POST
ACTION=″mailto:rtm@umail.uz″>.

В форме вопросник организуется непарным тэгом

<**INPUT**>, вместе с параметром **NAME**. Для ввода ответа на вопрос, заданный с помощью этого тэга, параметр создает текстовое поле (строку для ответа). Число символов в строке для ответов определяется параметром **SIZE** (размер). Вопрос вводится как обычный текст. Например:

<P> Ваша фамилия:

<INPUT NAME = "место для ввода фамилии" SIZE=25>

Эти тэги в форме выводят текст «Ваша фамилия:» и создают текстовое поле под названием «место для ввода фамилии», куда можно вводить 25 символов. Иногда ответ, вводимый в некоторый раздел формы, может не поместиться в одну строку. Например, если в форме есть раздел «Примечание», обычно для него выделяется текстовое поле, состоящее из нескольких строк. Для этого используется парный тэг **<TEXTAREA>**. В состав этого тэга входят параметры, определяющие имя текстового поля **(NAME)**, число строк **(ROWS)** и число столбцов **(COLS)**. Например,

<P> Примечание:

<TEXTAREA NAME="Примечание" ROWS=4 COLS=40>

</TEXTAREA>.

Эти тэги в форме выводят слово «Примечание» и создают текстовое поле «Примечание» с 4 строками и 40 столбцами (т.е. 4 строки с 40 символами).

Есть такие вопросы, для которых нужно выбрать только один среди нескольких ответов. Например, на вопрос об образовании выбирается один из ответов «Начальное», «Среднее», «Средне-специальное» или «Высшее». Ответы на такие вопросы можно заранее вводить в форму. Обычно, перед такими ответами ставят кружок, и с помощью мышки перед нужным ответом выбирается кружок. Для организации в форме такого вопросника на веб-странице вместе с тэгом <INPUT> используется параметр NAME и параметр TYPE (тип) со значением RADIO (направить).

А отправленный к вам (или на веб-сервер) ответ является значением параметра VALUE (значение). Например:

<P> Образование:

<INPUT TYPE=radio NAME="Образование" value ="Начальное"> Начальное

<INPUT TYPE=radio NAME="Образование" value ="Среднее"> Среднее

<INPUT TYPE=radio NAME=″Образование″ value =″Средне-специальное″> Средне-специальное

<INPUT TYPE=radio NAME=″Образование″ value ="Высшее″> Высшее.

Здесь параметр TYPE=radio создает кружок в форме; слово «Образование» записанное после NAME=, является именем поля и не отражается на экране; слово «Начальное» записанное после VALUE= отправляется на веб-сервер, а слово «Начальное», записанное после него, отражается на экране возле кружочка. А тэг
 переводит следующий текст на новую строку. Иногда потребуется выбрать несколько ответов среди предложенных ответов. В этом случае вместо параметра RADIO тэга TYPE используется параметр **CHECKBOX** (место выбора). Тогда в форме вместо кружочка появляется квадратик.

Например, ответы на вопрос об освоенных языках в HTML-документе можно выразить следующим образом:

<P> Освоенные языки:

<INPUT TYPE=checkbox NAME=″Язык″ value= ″Uzbekish″>Узбекский

<INPUT TYPE=checkbox NAME=″Язык″ value=″Russian″> Русский

<INPUT TYPE=checkbox NAME=″Язык″ value=″English″> Английский

<INPUT TYPE=checkbox NAME=″Язык″ value=″German″> Немецкий

С помощью рассмотренных тэгов вы можете создать несложные формы. Если готовую форму загрузите в сеть Интернет, то ее увидят миллионы. Но ответы не дойдут до вас. Чтобы ответы вернулись к вам, в тэге <INPUT> применяется параметр TYPE со значением **SUBMIT** (представлять). Для очистки текстового поля в тэге <INPUT> применяется параметр TYPE со значением **RESET** (обновить). Если в этом тэге использовать параметр **VALUE**, то браузер на веб-странице создает кнопку. Например, тэг

<INPUT TYPE=″submit″ VALUE=″Отправить форму″> на веб-странице создает кнопку с надписью «Отправить форму». Если выбрать эту кнопку, то вся введенная информация отправляется в нужный адрес. А тэг

<INPUT TYPE=″reset″ VALUE=″Обновить форму″>

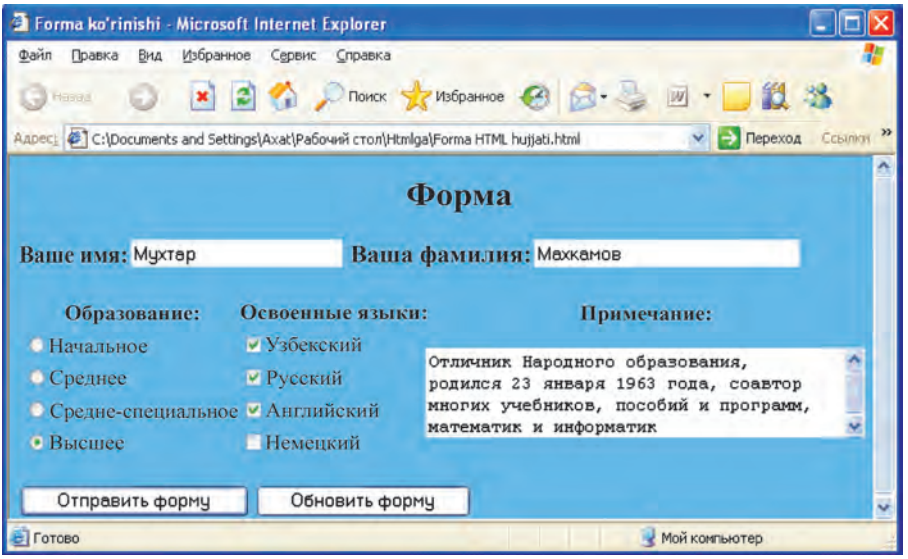
создает на веб-странице кнопку с надписью «Обновить форму». Если выбрать эту кнопку, то вся введенная информация «исчезнет», т.е. форма возвращается в исходное положение.

Внизу приведены HTML-документ формы и соответствующая веб-страница:

```
<HTML>
<title>Вид формы</title>
<BODY BGCOLOR="#55AAFF"><H2 ALIGN="center">Форма</H2>
<FORM METHOD=POST ACTION="mailto:rtm@umail.uz">
<FONT SIZE=4>Ваше имя: <INPUT NAME="имя" SIZE=26>
Ваша фамилия: <INPUT NAME="фамилия" SIZE=30><P>
<table>
<TR><TH>Образование:<BR></TH><TH>Освоенные языки:<B
R></TH><TH>Примечание:</TH></TR> <TR><TD><INPUT TYPE=radio
NAME="Образование" value="Начальное">
Начальное</TD>
<TD><INPUT TYPE=checkbox NAME="Язык" value="Узбекский">
Узбекский</TD>
<TD ROWSPAN=4><TEXTAREA ROWS=4 COLS=40></TEXTAREA></TD>
</TR>
<TR><TD><INPUT TYPE=radio NAME="Образование"
value="Среднее">Среднее</TD>
<TD><INPUT TYPE=checkbox NAME="Язык" value="Русский">
Русский</TD></TR>
<TR><TD><INPUT TYPE=radio NAME="Образование"
value="Средне-специальное">Средне-специальное</TD>
<TD><INPUT TYPE=checkbox NAME="Язык" value="Английский">
Английский</TD></TR>
<TR><TD><INPUT TYPE=radio NAME="Образование"
value="Высшее"> Oliy</TD>
<TD><INPUT TYPE=checkbox NAME="Язык" value="Немецкий">
Немецкий</TD></TR>
</table>
<P><INPUT TYPE="submit" value="Отправить форму"><INPUT
TYPE="reset" value="Обновить форму"></FONT></FORM> </BODY>
</HTML>
```

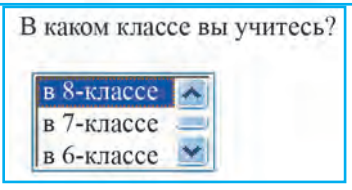
В этой форме использованы возможности добавления таблиц. Поэтому обе кнопки выбора расположены рядом.

При создании формы возможность выбора можно осуществить с помощью парного тэга <SELECT>. В этом случае данные для выбора отражаются в виде списка со скроллингом. Элементы списка вводятся с помощью непарного тэга <**OPTION**>. Тэг SELECT имеет параметры NAME, SIZE,



MULTIPLE. Их действия можно понять на следующем примере.

```
<html>
<title>Select</title>В каком классе вы учитесь?<br>
<SELECT name="" size=3>
<OPTION value="9">в 9-классе <OPTION value="8">в 8-классе
<OPTION value="7">в 7-классе <OPTION value="6">в 6-классе
<OPTION value="5">в 5-классе </select>
</html>
```



Как видите, часть выбора ответа формы занимает очень мало места. В этом примере на основе выбора в нужный адрес отправляется только одно значение — «8».

Если необходимо выбрать несколько значений, то достаточно добавить параметр MULTIPLE, который не требует ввода значений.

Вопросы и задания

1. *Расскажите об известных вам формах.*
2. *Для чего нужны формы?*
3. *Как организуется форма на веб-странице?*

4. *Расскажите об отправке введенных в форму данных.*
5. *Как определяется текстовое поле в форме?*
6. *Каким образом можно организовать текстовое поле с несколькими строками?*
7. *Расскажите о выборе ответов, заранее введенных в форму.*

Упражнения

1. Создайте форму-вопросник для участия в конкурсе веб-страниц по теме «Почему я люблю родной Узбекистан?».
2. Создайте форму под названием «Приглашение дружить». В ней должна содержаться информация на основе интересующих вас вопросов.
3. Создайте форму под названием «Библиотечная анкета».

Урок 63. Повторение тем «Переход (гиперссылка) на веб-странице» и «Формы»

1. Создайте главную веб-страницу под названием «Мои веб-страницы». Организуйте переход к веб-страницам, созданным вами, и обратно на главную веб-страницу с помощью текста и рисунка.
2. Создайте форму под названием «Об учебнике по информатике». Цель формы – сбор информации об учебнике.
3. Создайте веб-страницу под названием «Школа, информатика и я», отражающую ваши практические работы с 5 по 9-классы.

Урок 64. Интерактивные веб-страницы

Веб-сайты являются объединением нескольких веб-страниц, связанных через гиперссылки (hyperlink), и их можно условно делить на следующие два типа:

статические	динамические

Статические веб-сайты – это объединение не меняющихся веб-страниц, содержащих тексты, рисунки и другую информацию, и кодированных с их взаимосвязями. Они составляются из стандартных документов и данных, интересных для пользователя. Если требуется их обновить или добавить дополнительные данные, то каждый раз придется менять программный код. А это требует много времени и труда, и в связи с увеличением числа веб-страниц тяжелее становится управлять веб-сайтом. Следует отметить, что в эпоху основания Интернета все веб-сайты были статичес-

ками. В настоящее время все веб-сайты создаются в динамическом виде. Динамические веб-сайты состоят из веб-страниц, на которых разные части информации меняются независимо друг от друга в процессе исполнения запроса пользователя. В динамических веб-сайтах работа с информацией организуется по запросу пользователя на основе данных, хранящихся в базе данных сервера.

Отличие статических и динамических веб-сайтов можно увидеть на следующем примере (**my.gov.uz** – единый портал интерактивных государственных услуг). Показанные на рисунке веб-страницы принадлежат одному веб-сайту, и по запросу пользователя вместо первой веб-страницы открылась вторая веб-страница.

Если бы данный веб-сайт относился к статическому типу, то одна и та же информация, содержащаяся на обоих веб-страницах, кодировалась бы отдельно для каждой веб-страницы (повторяющаяся информация указана цифрами). Так как веб-сайт относится к динамическому типу, то первая веб-страница переходит на вторую с заменой соответствующей части по специальному сценарию. Код оставшихся без изменений частей для обеих веб-страниц будет общим. Следовательно, если веб-сайт состоит из многих схожих между собой веб-страниц, то, несомненно, очень важно, чтобы он относился к динамическому типу. Одним из основных удобств динамических веб-сайтов является легкость управления информационными ресурсами через окно администратора.

Динамические сайты очень удобны для применения интерактивных (англ. interaction – взаимодействие) технологий, и веб-сайты, использующие такие технологии, называются **интерактивными веб-сайтами**. В настоящее время особое внимание обращают на свойство интерактивности веб-сайтов. Но, веб-сайты с флеш-анимациями или мультимедийными ресурсами во многих случаях ошибочно рассматривают как интерактивные веб-сайты.

Интерактивные веб-сайты не ограничиваются только предоставлением возможности просмотра или ознакомления, еще в них предусмотрена возможность регистрации, отправки и приема сообщений, проведения онлайн (англ. online – диалог, в сети) опросов, получения информации по заказу, «диалог» пользователя через разные счетчики и другие элементы. Кроме этого многие интерактивные веб-сайты предоставляют возможность в режиме реального времени проводить онлайн-беседы между пользователем и администрацией сайта, напрямую связываться через онлайн-чаты (англ. chatter – разговаривать).

Для реализации интерактивных «свойств» веб-сайтов применяются специальные программные коды – серверные скрипты. Эти скрипты обеспечивают отражение в веб-странице полученных данных от пользователя после обработки на сервере. Обычно, браузер читает html-файл, если этот html-файл содержит скрипты, то сначала выполняются действия по предусмотренному этими скрип-

тами сценарию, после чего полученные данные отражаются браузером. Из-за того, что скрипты выполняются на сервере, и потом только результат отправляется в браузер, исходные коды серверных скриптов не отражаются браузером.

С помощью серверных скриптов можно:

• оперативно добавить и изменить любую информацию;

• получить ответ на запрос пользователя или отправить данные;

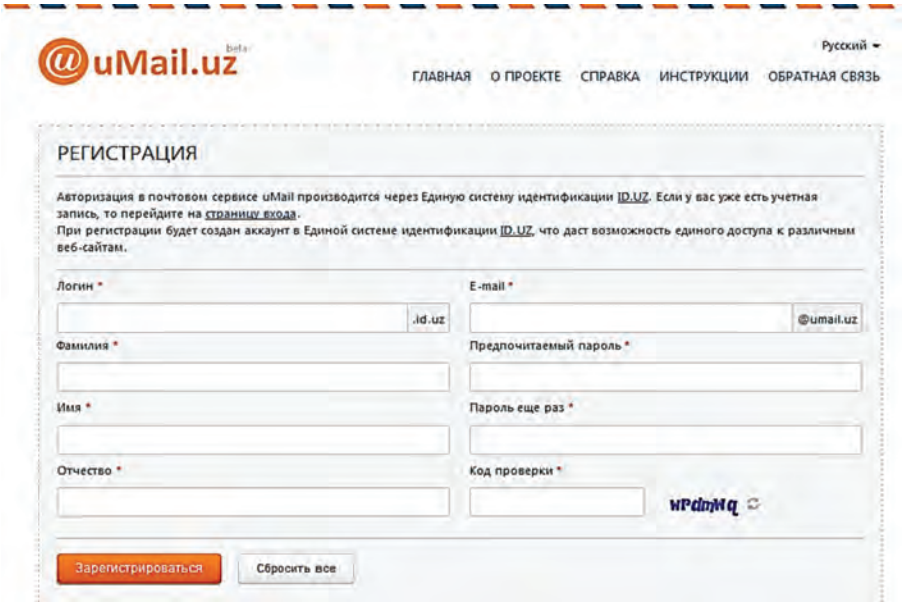
• войти в любую базу данных или информации;

• по усмотрению пользователя изменить или настроить веб-страницу.

Существует ряд правил для интерактивных веб-страниц, перечислим основные из них:

– почти все веб-страницы веб-сайта **генерируются** серверными программами и обрабатываются;

– данные для генерации веб-страниц берутся из соответствующих **баз данных**. Базы данных в специальных компьютерах сохраняются в различных видах;



– очень часто используется **ограничение доступа** к веб-сайту. Ограничения доступа могут быть разными для разных клиентов. Обычные посетители могут только просмат-

ривать данные, в то время как другие посетители имеют возможность вносить в них изменения. В этом случае используются элементы идентификации (латин. identifico – приравнивать, считать равным), то есть логин (имя идентификатора пользователя) и пароль (франц. слово) (см. рисунок ниже);

– веб-сайт содержит поисковую систему данных.

Динамические веб-сайты разрабатываются с помощью широко распространённых и всеми принятых программных систем JavaScript, PHP, Perl, создаются серверными скриптами. Эти языки программирования позволяют создавать веб-сайты любой сложности в полной мере. Но, для выполнения этих работ требуются глубокие навыки и поэтому они выполняются программистами.

Вопросы и задания

1. *На какие типы делятся веб-страницы исходя из технологий?*
2. *Расскажите об интерактивной веб-странице и приведите примеры.*
3. *Расскажите о веб-страницах, в которых должны применяться логин и пароль.*
4. *В какой веб-странице вы регистрируете свою электронную почту?*
5. *В каких электронных учебниках и пособиях в вашем компьютере есть интерактивная веб-страница?*

Уроки 65—66. Задания для самостоятельной работы

С помощью тэгов и символов, описанных внизу, введите изменения в свои веб-страницы:

1. Изменение формата шрифта:

В программе MS Word	**Тэг HTML**	**Пример в HTML**	**На веб-странице**
надстрочный, то есть x^2	Парный тэг <SUP>	ma^{kt}ab	ma^{kt}ab
подстрочный, то есть x_2	Парный тэг <SUB>	ma_{kt}ab	ma$_{kt}$ab
зачеркнутый, то есть abc	Один из парных тэгов <S>, <STRIKE>, <DEL>	~~maktab~~ <s>maktab</s> <strike>maktab</strike>	maktab
Примечание	Непарный тэг <!>	<! школа>	Не отражается

2. Специальные знаки:

В программе MS Word	**Запись в HTML**	**Пример в HTML**	**На веб-странице**
<	&LT	&lt	<
>	&GT	&gt	>
&	&AMP	&amp	&
"	&QUOT	&quot9-sкласс&quot	"9 класс"
Твердый пробел		5 балов	5 балов
©	&COPY	&copy	©

3. Эффекты:

Запись в HTML	**Пример в HTML**	**На веб-странице**
<ACRONYM>	<acronym title= "274-школа">9-класс< /acronym>	При направлении указателя мышки к записи «9-класс» 274-школа
<HR>	<hr align=center size= color= red width= 60% noshade>	Рисует линию красного цвета толщиной 4 пиксель, занимавшего 60 процентов веб-страницы, выровненный по центру, параметр NOSHADE удаляет выпуклость линии
<MARQUEE>	"<marquee behavior= "alternate" width=60% height=30% bgcolor= blue> <Font size= 7 color= "white"> ДВИЖЕНИЕ </marquee>"	Внутри прямоугольника синего цвета, занимавшего по горизонтали 60 процентов и по вертикали 30 процентов веб-страницы, передвигается слово «Движение» белого цвета с размером шрифта 7

Уроки 67—68. Задания для повторения

1. Составьте программу, которая покажет сравнение количества символов, встречающихся в заданном тексте A, с количеством пробелов текста A. Например: «a−21» > «probel − 7».

2. Составьте программу следующего содержания: для заданного N ($1<N<15$) она рисует N штук окружностей с центром в центре экрана.

3. Составьте программу, которая находит остаток от деления заданного N ($N>21$) разрядного числа на 2 и на 3.

4. Составьте программу, которая находит количество всех делителей числа N.

5. Создайте веб-страницу под названием «Прощай школа!».

ИСПОЛЬЗОВАННЫЕ ОСНОВНЫЕ ИСТОЧНИКИ

1. *Б. Болтаев, А. Абдукадыров, Н. Тайлаков, М. Махкамов, А. Азаматов, С. Хафизов.* Информатика и основы вычислительной техники. 9-класс. – Т.: Чулпан, 2006.

2. *Б. Болтаев, М. Махкамов, А. Азаматов.* Программирование на языке Паскаль. Методическое пособие. – Т.: 2007.

3. *Б. Болтаев, А. Азаматов, Ш. Хидиров, Б. Хуррамов, Г. Ишанходжаева.* Методы решения задач по алгоритмизации и на языке программирования Паскаль. Методическое пособие для учителей. – Т.: Издательство «NIHOL», 2012.

4. *Л.Л. Босова, А.Ю. Босова.* Информатика, 7–9. Издательство «БИНОМ», – М., 2013.

5. *Л.З. Шауцукова.* Информатика, 10–11. Издательство «Просвещение», – М., 2000.

6. *А.Г. Кулаков, С.К. Ландо, А.Л. Семенов, А.Х. Шень.* Алгоритмика, V–VII классы. – М., Дрофа, 1997.

7. *А.Н. Степанов.* Информатика, Учебник для вузов. Санкт-Петербург: Питер, 2006.

Примечание. Полный список источников по дате и терминов в учебнике утвержден и рекомендован решением от 12 марта 2015 года научно методического совета по предмету Информатика при Республиканском центре образования.

Вышеупомянутый список внесен в веб-сайт Республиканского центра образования (rtm.uz).

ОГЛАВЛЕНИЕ

ГЛАВА I. ОСНОВЫ АЛГОРИТМИЗАЦИИ

Урок 1. Этапы решения задач на компьютере....3
Урок 2. Модель и ее типы....7
Урок 3. Повторение тем «Этапы решения задач на компьютере» и «Типы моделей»....12
Урок 4. Понятие алгоритма....13
Урок 5. Основные свойства алгоритма....17
Урок 6. Повторение тем «Понятие алгоритма» и «Основные свойства алгоритма»....19
Урок 7. Способы представления алгоритмов....20
Урок 8. Практическое занятие по теме «Способы представления алгоритмов»....23
Урок 9. Основные типы алгоритмов....24
Урок 10. Практическое задание по базовым структурам алгоритма....28
Урок 11. Задания для повторения....31

ГЛАВА II. ОСНОВЫ ПРОГРАММИРОВАНИЯ

Урок 12. Программа и языки программирования....32
Урок 13. Интегрированная среда Turbo Pascal 7.0....35
Урок 14. Алфавит языка Паскаль и его структура....38
Урок 15. Постоянные и переменные величины....43
Урок 16. Повторение темы «Постоянные и переменные величины»....47
Урок 17. Табличные величины....47
Урок 18. Повторение темы «Табличные величины»....51
Урок 19. Стандартные функции и процедуры, алгебраические выражения....52
Урок 20. Повторение темы «Стандартные функции и процедуры, алгебраические выражения»....57
Урок 21. Операторы присваивания и вывода информации на экран....57
Урок 22. Повторение темы «Операторы присваивания и вывода информации на экран»....61
Урок 23. Оператор ввода данных в память компьютера в диалоговом режиме....62
Урок 24. Повторение темы «Операторы ввода данных в память компьютера в диалоговом режиме»....65
Урок 25. Работа с экраном в текстовом режиме....66
Урок 26. Повторение темы «Работа с экраном в текстовом режиме»....69
Урок 27. Составление линейных программ....70
Урок 28. Повторение темы «Составление линейных программ»....73

Урок 29. Операторы перехода и ветвления....73
Урок 30. Повторение темы «Операторы перехода и ветвления»....77
Урок 31. Составление программ со структурой ветвления....77
Урок 32. Повторение темы «Составление программ со структурой ветвления»....81
Урок 33. Оператор цикла с параметром....81
Урок 34. Повторение темы «Оператор цикла с параметром»....85
Урок 35. Операторы цикла по условию....85
Урок 36. Повторение темы «Операторы цикла по условию»....88
Урок 37. Задания для повторения....89
Урок 38. Работа с символьными и строковыми величинами....90
Урок 39. Повторение темы «Работа с символьными и строковыми величинами»....95
Урок 40. Перевод экрана в графический режим....95
Урок 41. Повторение темы «Перевод экрана в графический режим»....99
Урок 42. Возможности рисования фигур на языке Паскаль....100
Урок 43. Повторение урока «Возможности рисования фигур на языке Паскаль»....104
Урок 44. Работа с файлами....104
Урок 45. Повторение темы «Работа с файлами»....110
Урок 46. Процедуры и функции....110
Урок 47. Повторение темы «Процедуры и функции»....114
Уроки 48—49. Задания для повторения....114

ГЛАВА III. СОЗДАНИЕ WEB-СТРАНИЦ

Урок 50. Понятие о HTML....115
Урок 51. Ввод текста на веб-страницу....118
Урок 52. Повторение темы «Ввод текста на веб-страницу»....123
Урок 53. Размер и цвет шрифта, фон веб-страницы....124
Урок 54. Повторение темы «Размер и цвет шрифта, фон веб-страницы» 127
Урок 55. Графика на веб-странице....128
Уроки 56—57. Повторение темы «Графика на веб-странице»....133
Урок 58. Вставка списка на веб-страницу....133
Урок 59. Вставка таблиц на веб-страницу....137
Урок 60. Повторение тем «Вставка списка на веб-страницу» и «Вставка таблиц на веб-страницу»....141
Урок 61. Переход (гиперссылка) на веб-странице....142
Урок 62. Формы 145
Урок 63. Повторение тем «Переход (гиперссылка) на веб-странице» и «Формы»....150
Урок 64. Интерактивные веб-страницы....150
Уроки 65—66. Задания для самостоятельной работы....154
Уроки 67—68. Задания для повторения....155
Использованные основные источники....156

Bahodir Jalolovich Boltayev
Axat Raxmatovich Azamatov
Abror Davlatmirzayevich Asqarov
Muxtor Qurbonovich Sodiqov
Gulnoza Axatovna Azamatova

INFORMATIKA
VA HISOBLASH TEXNIKASI ASOSLARI

(Rus tilida)

2-nashri

Umumiy o‘rta ta’lim maktablarining 9-sinfi uchun darslik

Перевод ***А.Р. Азаматова***

Редактор ***Георгий Хубларов***
Художественный редактор ***Сардор Курбанов***
Технический редактор ***Елена Толочко***
Корректор ***Умида Раджабова***
Оператор ***Гульчехра Азизова***

Номер лицензии AI № 163. 09.11.2009. Подписано в печать 26 июня 2015 года. Формат 60×90$^{1}/_{16}$. Гарнитура Таймс. Кегель 11. Офсетная печать. Условных печатных листов 10,0. Учетно-издательских листов 8,92. Тираж 38287 экз. Договор № 29–2015. Заказ № 233.

Издательско-полиграфический творческий дом имени Чулпана Узбекского агентства печати и информации. 100129. г. Ташкент, ул. Навои, 30.
Телефон (371) 244-10-45. Факс (371) 244-58-55.

Отпечатано совместно в типографиях издательско-полиграфических творческих домов им. Гафура Гуляма и «O‘zbekiston» при Узбекском агентстве по печати и информации. 100128, г. Ташкент, ул. Лабзак, 86/100129, г. Ташкент, ул. Навои, 30.

Б 79 **Информатика и основы вычислительной техники**:Учебник для 9-го класса общеобразовательных средних школ /Б. Ж. Болтаев [и др.]; отв. ред. Н. Тайлаков. – Т.: ИПТД имени Чулпана, 2015. – 160 с.

1. Болтаев, Б. Ж.

ISBN 978-9943-05-745-6

УДК: 372.8:004(075)
ББК 32.81(5У)я721

Сведения о состоянии арендного учебника

№	Имя, фамилия ученика	Учебный год	Состояние учебника при получении	Подпись классного руководителя	Состояние учебника при сдаче	Подпись классного руководителя
1						
2						
3						
4						
5						
6						

Таблица заполняется классным руководителем при передаче учебника в аренду и возвращении назад в конце учебного года. При заполнении таблицы используются следующие оценочные критерии

Новый учебник	Состояние учебника при первой передаче в аренду
Хорошо	Обложка цела, не оторвана от блока книги. Все страницы в наличии, не порваны, не выпадают из блока, на страницах нет записей и помарок.
Удовлетворительно	Обложка несколько отходит от блока, слегка помята, испачкана, края потёрты; удовлетворительно восстановлена пользователем. Некоторые страницы исчерчены, выпавшие страницы приклеены. Учебник отреставрирован.
Неудовлетворительно	Обложка испачкана, порвана, оторвана от блока книги или совсем отсутствует. Страницы порваны, исчерчены, в помарках, некоторых страниц не хватает. Учебник не пригоден к восстановлению.