8 класс
Билеты по информатике с ответами

Билет № 1

Понятие алгоритма: линейный, разветвленный, циклический. Исполнитель алгоритма. Система команд исполнителя (на примере учебного исполнителя Кенгуру). Свойства алгоритма. Способы записи алгоритмов; блок-схемы.

.

✓	Под **алгоритмом** понимают постоянное и точное предписание (указание) исполнителю совершить определенную последовательность действий, направленных на достижение указанной цели или решение поставленной задачи

Слово алгоритм происходит от *algorithmi* – латинской формы написания имени великого математика IX в. Аль Хорезми, который сформулировал правила выполнения арифметических действий. Первоначально под алгоритмами и понимали только правила выполнения четырех арифметических действий над многозначными числами. В дальнейшем это понятие стали использовать вообще для обозначения последовательности действий, приводящих к решению поставленной задачи.

Рассмотрим пример алгоритма для нахождения середины отрезка при помощи циркуля и линейки.

Алгоритм деления отрезка АВ пополам:
1) поставить ножку циркуля в точку А;
2) установить раствор циркуля равным длине отрезка АВ;
3) провести окружность;
4) поставить ножку циркуля в точку В;
5) провести окружность;
6) через точки пересечения окружностей провести прямую;
7) отметить точку пересечения этой прямой с отрезком АВ.

Анализ примеров различных алгоритмов показывает, что запись алгоритма распадается на отдельные указания исполнителю выполнить некоторое законченное действие. Каждое такое указание называется ***командой***. Команды алгоритма выполняются одна за другой. После каждого шага исполнения алгоритма точно известно, какая команда должна выполнятся следующей. Совокупность команд, которые могут быть выполнены исполнителем, называется ***системой команд исполнителя.***

Свойства алгоритмов:

1. Поочередное выполнение команд алгоритма за конечное число шагов приводит к решению задачи, к достижению цели. Разделение выполнения решения задачи на отдельные операции (выполняемые исполнителем по определенным командам) – важное свойство алгоритмов, называемое ***дискретностью***.
2. Каждый алгоритм строится в расчете на некоторого исполнителя. Для того чтобы исполнитель мог решить задачу по заданному алгоритму, необходимо, чтобы он был в состоянии понять и выполнить каждое действие, предписываемое командами алгоритма. Такое свойство алгоритмов называется ***определенностью (****или* ***точностью)*** *алгоритма*. (Например, в алгоритме указано, что надо взять 3—4 стакана муки. Какие стаканы, что значит 3—4, какой муки?)
3. Еще одно важное требование, предъявляемое к алгоритмам, - ***результативность (****или* ***конечность)*** *алгоритма*. Оно означает, что исполнение алгоритма должно закончиться за конечное число шагов.
4. ***Универсальность***. Алгоритм должен быть составлен так, чтобы им мог воспользоваться любой исполнитель для решения аналогичной задачи. (Например, правила сложения и умножения чисел годятся для любых чисел, а не для каких-то конкретных.)

Таким образом, выполняя алгоритм, исполнитель может не вникать в смысл того, что он делает, и вместе с тем получать нужный результат. В таком случае говорят, что исполнитель действует ***формально***, т.е. отвлекается от содержания поставленной задачи и только строго выполняет некоторые правила, инструкции.

Способы задания алгоритма:

- словесный, (недостаток–многословность, возможна неоднозначность–«он встретил ее на поле с цветами»),
- табличный (физика, химия и т. д.),
- графический (блок-схемы).

Графическая форма представления алгоритма называется **блок-схемой**

Условные обозначения

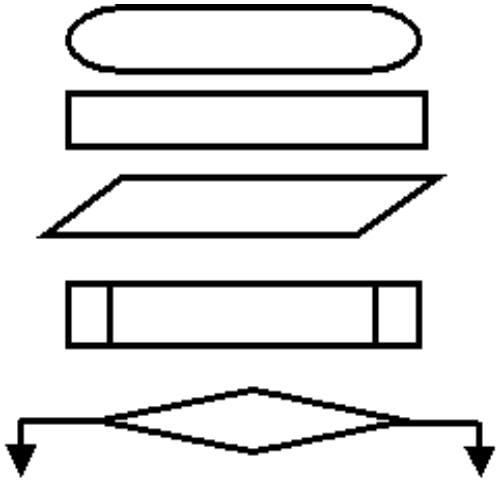
Начало - конец

Процесс

Ввод-вывод

Типовой процесс

Решение (условие)



Базовые алгоритмические структуры

Линейный **Ветвление** **Повторение (цикл)**

Разберем выполнение команд учебным исполнителем Кенгуру:

Команды **линейного алгоритма** в программе «Кенгу»

- Шаг
- Поворот на 90 градусов (против часовой стрелки)
- Прыжок

Команда **алгоритма ветвления на примере** рисования рамки листа **:**

- Если впереди не край, то
- Шаг
- Иначе поворот
- Конец ветвления

Команда циклического алгоритма на примере рисования прямой линии:

- Пока впереди не край, повторять
- шаг
- Конец цикла

Билет № 2

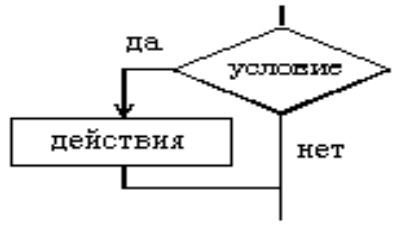
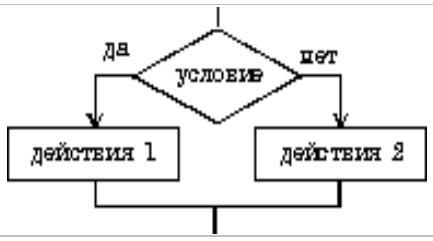
Основные алгоритмические структуры: следование, ветвление, цикл; изображение на блок-схемах.

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в разветвляющиеся алгоритмы входит условие, в зависимости от выполнения или невыполнения которого выполняется та или иная последовательность команд (серий).

В качестве условия в разветвляющемся алгоритме может быть использовано любое понятное исполнителю утверждения, которое может соблюдаться (быть истинно) или не соблюдаться (быть ложно). Такое

утверждение может быть выражено как словами, так и формулой. Таким образом, **команда ветвления** состоит из **условия** и **двух последовательностей команд**.

Структура ветвление существует в основных вариантах:

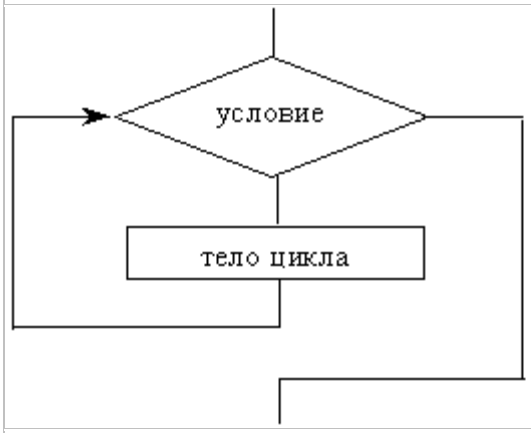
Язык блок-схем	Язык ТурбоПаскаль
1. если - то	
	**IF** условие **THEN** действия
2. если - то - иначе	
	**IF** условие **THEN** действие 1 **ELSE** действие 2

В циклические алгоритмы входит **последовательность команд , выполняемая многократно**. Такая последовательность команд называется **телом цикла.**

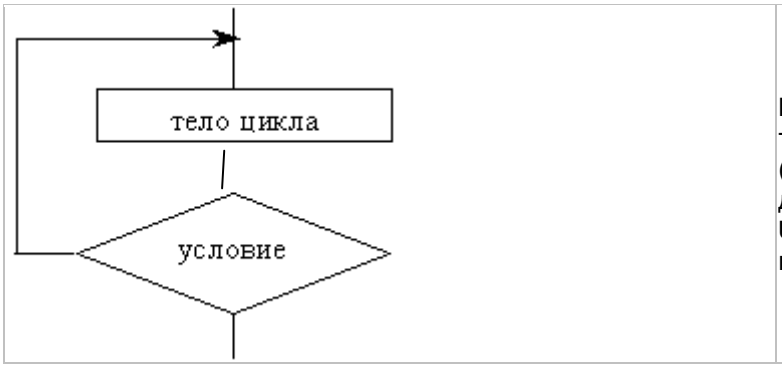
В **циклах типа пока** тело цикла выполняется до тех пор, пока **выполняется условие**. Выполнение таких циклов происходит следующим образом:
- пока условие справедливо (истинно), выполняется тело цикла,
- когда условие становится несправедливым, выполнение цикла прекращается.

Основные разновидности циклов представлены в таблице.

Язык блок-схем	Язык ТурбоПаскаль
1. Цикл с предусловием. (Цикл пока) Предписывает выполнение тела цикла до тех пор, пока выполняется условие, записанное после слова пока	

	**WHILE** условие **DO**
	Begin
	тело цикла (последовательность действий)
	End

2. Цикл с постусловием (Цикл до) Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне	



Repeat
тело цикла
(последовательность действий)
Until условие проверки цикла на окончание

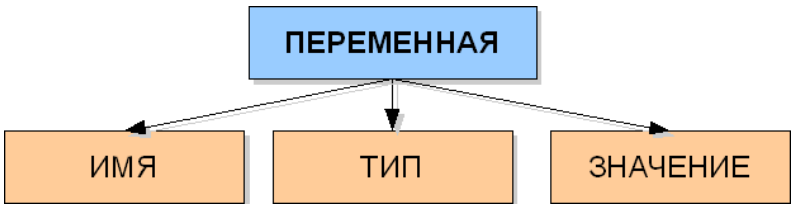
Билет № 3

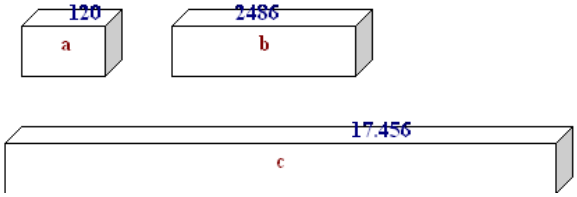
Величины: константы, переменные, типы величин. Присваивание, ввод и вывод величин. Линейные алгоритмы работы с величинами.

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

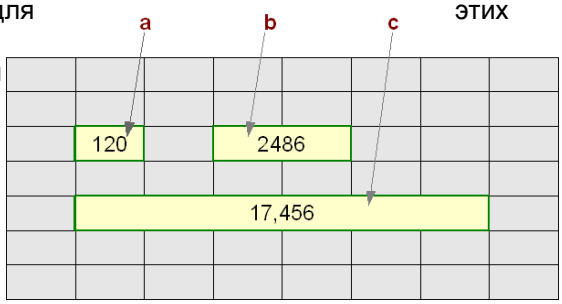
Величины в программировании, как и в математике, делятся на переменные и константы. Значение константы остается неизменной в течении всей программы, значение переменной величины может изменяться.

У каждой переменной есть имя, тип и текущее значение. Имена переменных называют идентификаторами (от глагола «идентифицировать», что значит «обозначать», «символизировать»). В качестве имен переменных могут быть буквы, цифры и другие знаки. Причем может быть не одна буква, а несколько. Примеры идентификаторов: a, b5, x, y, x2, summa, bukva10...





Существуют три основных типа величин, с которыми работает компьютер: **числовой, символьный и логический**. Тип данных характеризует внутреннее представление, множество допустимых значений для этих данных, а также совокупность операций над ними. В зависимости от типа переменной в памяти компьютера будет выделена определенная область.



Наглядно переменную можно представить как коробочку, в которую можно положить на хранение что-либо. Имя переменной – это надпись на коробочке, значение – это то, что хранится в ней в данный момент, а тип переменной говорит о том, что допустимо класть в эту коробочку.

Всякий алгоритм строится исходя из системы команд исполнителя, для которого он предназначен.Независимо от того, на каком языке программирования будет написана программа, алгоритм работы с величинами, обычно, составляется из следующих команд:

- ✓ присваивание;
- ✓ ввод;
- ✓ вывод;

Значения переменным задаются с помощью оператора присваивания. Команда присваивания – одна из основных команд в алгоритмах работы с величинами. При присваивании переменной какого-либо значения старое значение переменной стирается и она получает новое значение.

В языках программирования команда присваивания обычно обозначается либо «:=» (двоеточие и равно), Значок «:=» читается «присвоить». Например:

z := x + y

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

Если до выполнения этой команды содержимое ячеек, соответствующих переменным x, y, z, было таким:

x	y	z
2	3	-

то после выполнения команды z := x + y оно станет следующим:

x	y	z
2	3	5

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

Если слева от знака присваивания стоит числовая переменная, а справа – математическое выражение, то такую команду называют арифметической командой присваивания, а выражение – арифметическим.

В частном случае арифметическое выражение может быть представлено одной переменной или одной константой.

Например:

x := 7

a := b + 10

c := x

Значения переменных, являющихся исходными данными решаемой задачи, как правило, задаются вводом. На современных компьютерах ввод чаще всего выполнятся в режиме диалога с пользователем. По команде ввода компьютер прерывает выполнение программы и ждет действий пользователя. Пользователь должен набрать на клавиатуре вводимые значения переменных и нажать клавишу <ВВОД>. Введенные значения присвоятся соответствующим переменным из списка ввода, и выполнение программы продолжится.

Команд ввода в описаниях алгоритмов обычно выглядит так:

ввод (<список переменных>)

Вот схема выполнения приведенной выше команды.

1. Память до выполнения команды:

a	b	c
-	-	-

2. Компьютер получил команду ввод (a, в, c), прервал свою работу и ждет действий пользователя.

3. Пользователь набирает на клавиатуре:

1 3 5

и нажимает клавишу <ВВОД> (<Enter>).

4. Память после выполнения команды:

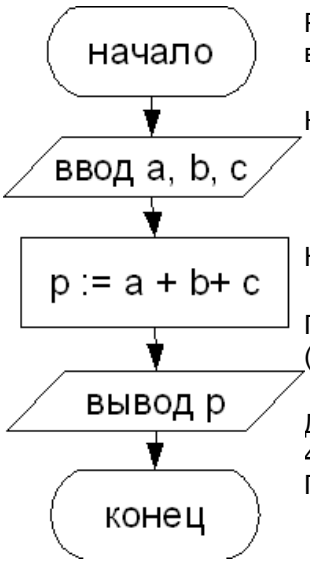
a	b	c
1	3	5

При выполнении пункта 3 вводимые числа должны быть отделены друг от друга какими-нибудь разделителями. Обычно это пробелы.

Следовательно, можно сделать вывод:

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

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



Результаты решения задачи сообщаются компьютером пользователю путем выполнения команды вывода.

Команда вывода в описаниях алгоритмов обычно выглядит так:

вывод (<список вывода>)

Например: **вывод (x1, x2)**.

По этой команде значения переменных x1 и x2 будут вынесены на устройство вывода (чаще всего это экран).

Для примера составим алгоритм вычисления периметра треугольника. Нам потребуется 4 переменных для хранения значения длин сторон треугольника и его периметра. Периметр – это сумма всех сторон.

```
Алгоритм Вычисление периметра треугольника
 переменные a, b, c, p - целые
 начало
 ввод (a, b, c)
 p := a + b+ c
 вывод (p)
 конец
```

Сначала компьютер запросит значения переменных a, b, c у пользователя, затем произведет вычисления и выведет результат на экран.

Строка **переменные a, b, c, p - целые** - называется описанием переменных. Некоторые языки программирования требуют обязательного описания всех переменных до начала их использования в программе, некоторые – относятся более лояльно.

Полученный алгоритм имеет линейную структуру.

Билет № 4

Логические величины, операции, выражения. Логические выражения в качестве условий в ветвящихся и циклических алгоритмах.

Для того чтобы понять работу ветвящихся и циклических алгоритмов, рассмотрим понятие логического выражения. В некоторых случаях выбор варианта действий в программе должен зависеть от того, как соотносятся между собой значения каких-то переменных.
В результате сравнения значений двух выражений возможны два варианта ответа: сравнение истинно или ложно?
Например:

2+3 > 3+1 - да (истинно)
0 < -5 - нет (ложно)

Выражения такого вида мы будем называть логическими выражениями.
Логическое выражение, подобно математическому выражению, выполняется (вычисляется), но в результате получается не число, а логическое значение: истина (true) или ложь (false). Логическая величина – это всегда ответ на вопрос, истинно ли данное высказывание.
Нам известны шесть операций сравнения:

знак отношения	*операция отношения*
=	равно
<>	не равно
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

С помощью этих операций мы будем составлять логические выражения. Причём в выражениях не обязательно присутствуют только константы, но и переменные.

5 > 3

a < b

c <> 7

Как выполняются операции отношения для числовых величин понятно из математики. Как же сравниваются символьные величины? Отношение «равно» истинно для двух символьных величин, если их длинны одинаковы и все соответствующие символы совпадают. Следует учитывать, что пробел тоже символ.

Символьные величины можно сопоставлять и в отношениях >, <, >=, <=. Здесь упорядоченность слов (последовательности символов) определяется по алфавитному принципу.

«кот» = «кот»

«кот» < «лис»

«кот» > «дом»

Выражение, состоящее из одной логической величины или одного отношения, будем называть простым логическим выражением.

Часто встречаются задачи, в которых используются не отдельные условия, а совокупность связанных между собой условий (отношений). Например, в магазине вам нужно выбрать туфли, размер которых r = 45, цвет color = белый, цена price не более 400руб.

Другой пример: школьник выяснил, что сможет купить шоколадку, если она стоит 3руб. или 3руб. 50коп.

В первом примере мы имеем дело с тремя отношениями, связанными между собой союзом "и" и частицей "не", во втором - с двумя отношениями, связанными союзом "или". Подобные условия назовём *составными*, и для их обозначения в алгоритме договоримся использовать союзы "**и**", "**или**", "**не**", которые будем рассматривать как знаки логических операций, позволяющих из простых условий создавать составные, подобно тому, как из простых переменных и констант с помощью знаков +, - и т. д. можно создавать алгебраические выражения.

Так условия наших примеров в алгоритме могут выглядеть таким образом:

первое: (r = 45) **и** (color = белый) **и** (**не** (price>400))

второе: (цена=3) **или** (цена=3.5)

Выражение, содержащее логические операции, будем называть сложным логическим выражением.

Объединение двух (или нескольких) высказываний в одно с помощью союза «и» называется операцией *логического умножения* или *конъюнкцией*.

В результате логического умножения (конъюнкции) получается истина, если истинны все логические выражения.

Объединение двух (или нескольких) высказываний с по мощью союза «или» называется операцией *логического сложения* или *дизъюнкцией*.

В результате логического сложения (дизъюнкции) получается истина, если истинно хотя бы одно логическое выражения.

Присоединение частицы «не» к высказыванию называется операцией *логического отрицания* или *инверсией*.

Отрицание изменяет значение логической величины на противоположное: **не** истина = ложь; **не** ложь = истина.

Если в сложном логическом выражении имеется несколько логических операций, то возникает вопрос, в каком порядке их выполнит компьютер. По убыванию старшинства логические операции располагаются в таком порядке:

1. отрицание (**не**);
2. конъюнкция (**и**);
3. дизъюнкция (**или**).

В логических выражениях можно использовать круглые скобки. Так же как и в математических формулах, скобки влияют на последовательность выполнения операций. Если нет скобок, то операции выполняются в порядке их старшинства.

Пример. Пусть a, b, c – логические величины, которые имеют следующие значения: a = истина, b = ложь, c = истина. Необходимо определить результаты вычисления следующих логических выражений:

1. a **и** b
2. a **или** b
3. **не** a **или** b
4. a **и** b **или** c
5. a **или** b **и** c
6. **не** a **или** b **и** c
7. (a **или** b) **и** (c **или** b)
8. **не** (a **или** b) **и** (c **или** b)
9. **не** (a **и** b **и** c)

Получим в результате:

1. ложь

2. истина
3. ложь
4. истина
5. истина
6. ложь
7. истина
8. ложь
9. истина.

Пример. Составить алгоритм для вычисления: $x = \frac{\sqrt{4a - c}}{2a}$

Алгоритм Вычисление x
переменные a, c, x - вещественные
начало
ввод (a, c)
если (4*a – c >=0) и (a<>0) то
начало
x := корень(4*a – c)/(2*a)
вывод (x)
конец
иначе
вывод («нет решения»)
конец

Компьютер сначала проверит условие (4*a – c >=0) и (a<>0) и если оно окажется истинно, то вычислить x, иначе выведет сообщение «нет решения».

Пример. Составить алгоритм для вычисления суммы всех чисел от 1 до n.

Алгоритм Вычисление суммы чисел
переменные a, c, x - вещественные
начало
ввод (n)
x := 1
пока x<n повторять
начало
s := s + x
x := x +1
конец
вывод (s)
конец

До тех пор пока условие x<n будет истинно компьютер будет выполнять тело цикла – вычислять очередную сумму и увеличивать x на единицу.

Билет № 5

Представление о программировании: язык программирования Pascal; примеры несложных программ с линейной, ветвящейся и циклической структурой.

Назначение программирования - разработка программ управления компьютером с целью решения различных информационных задач. Для составления программ существуют разнообразные языки программирования.

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

В настоящее время существует много различных языков программирования: Кобол, С, Фортран, Visual Basic, Pascal и др.

Языки программирования - это формальные языки, специально созданные для общения человека с компьютером. Каждый язык программирования, равно как и "естественный" язык (русский, английский), имеет алфавит, словарный запас, свою грамматику, а также семантику.

- ✓ **Алфавит** - фиксированный для данного языка набор основных символов, допускаемых для составления текста программы на данном языке.
- ✓ **Синтаксис** - система правил, определяющих допустимые конструкции языка программирования из букв алфавита.
- ✓ **Семантика** - система правил однозначного толкования отдельных языковых конструкций, позволяющих воспроизвести процесс обработки данных.

При описании языка и его применении используют понятия языка. Понятие подразумевает некоторую синтаксическую конструкцию и определяемые ею свойства программных объектов или процесса обработки данных.

Система программирования - это программное обеспечение компьютера, предназначенное для разработки, отладки и исполнения программ, записанных на определенном языке программирования.

Система программирования предназначена для автоматизации разработки программного обеспечения. В состав системы программирования обязательно входят язык программирования, редактор для создания и исправления текстов программ и транслятор для перевода программ на язык машинных команд.

Центральный процессор компьютера может исполнять только команды на машинном языке, закодированные в двоичном алфавите. Программа, состоящая из таких команд, “понятна” компьютеру, но людям работать с последовательностями команд вида

011001010101010
111110001111100
101000010100101
..... …
101010010101001

совершенно неудобно.

Вскоре после появления первых компьютеров были разработаны специальные формальные языки – языки программирования высокого уровня, с более удобной для человека формой записи команд и не зависящие от особенностей архитектуры конкретного семейства компьютеров. Примерами таких языков являются Паскаль и Basic.

Для того, чтобы программа, написанная на языке программирования высокого уровня, могла быть выполнена компьютером, она должна быть переведена на язык его машинных команд. Это делается автоматически с помощью специальной программы-переводчика, называемой **транслятором**. Транслятор проверяет правильность записи команд на языке программирования высокого уровня и генерирует соответствующие последовательности команд на машинном языке.

Компилятор запоминает созданную для исходной программы последовательность машинных команд в специальном файле, но не дает команды компьютеру на их выполнение. Сохраненная компилятором в файле машинная программа может быть выполнена по команде пользователя в любое время. Это так называемый exe-ный файл.

Для созданных компилятором файлов машинных команд уже не требуется производить трансляцию, поэтому они выполняются быстрее, чем обрабатываемые интерпретатором исходные программы.

Язык программирования выского уровня Pascal

Язык программирования Паскаль (Pascal) с момента своего создания Никлаусом Виртом, швейцарским профессором, играет большую роль и в практическом программировании, и в его изучении. С непревзойденной четкостью в нем реализованы принципы структурного программирования. Паскаль стал первым языком, с которым знакомится большинство будущих программистов в мире.

Трансляторы программ, написанных на Паскале, разработаны на различных компьютеров и в настоящее время множество разновидностей. Они являются компиляторами, обрабатывающими разработанные программистами тексты программ.

Существует много версий языка Паскаль. Различия между ними порой весьма велики.

Любая Паскаль-программа является текстовым файлом с собственным именем и с расширением .pas.

Программа на языке Паскаль близка к своему виду к описанию алгоритма на Алгоритмическом языке.

Паскаль

```
Program Division;
var a,b,c,d,m,n: integer;
begin
readln (a,b,c,d); {ввод}
m:=a*d; {числитель}
n:=b*c; {знаменатель}
write (m,n); {вывод}
end.
```

Алгоритмический язык

```
алг деление дробей
цел a,b,c,d,m,n
нач
ввод a,b,c,d
m:=axd
n:=bxc
вывод m,n
кон
```

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

придуманное программистом.
Program <имя программы>;
Раздел описания переменных начинается со слова Var (variables - переменные), за которым идет список имен переменных через запятую. Тип указывается после двоеточия. В стандарте языка Паскаль существует два числовых типы данных: **вещественный и целый**. Слово **integer** обозначает целый тип (является идентификатором целого типа). Вещественный тип обозначается словом **real.** Например, раздел описания переменных может быть таким:
Var a, b: integer; c,d:real;
Идентификаторы переменных составляются из латинских букв и цифр; первым символом обязательно должна быть буква.
Раздел операторов (оператор - команда алгоритма, записанная на языке программирования) - основная часть программы. Начало и конец раздела операторов программы отмечается служебными словами begin (начало) и end (конец). В самом конце программы ставится точка.
begin
<операторы>
end.
Вывод результатов происходит по оператору write (write - писать) или writeln (writeln - писать в строку.
Арифметический оператор присваивания на Паскале имеет следующий формат:
<числовая переменная>:=
<арифметическое выражение>
Арифметическое выражение может содержать числовые константы и переменные, знаки арифметических операций, круглые скобки. Кроме того, в арифметических выражениях могут присутствовать функции.
Знаки основных арифметических операций записываются так:
+ сложение
- вычитание
* умножение
/ деление
Примеры программ на языке программирования Pascal

Линейная стуктура
Задача. Заданы длины двух катетов прямоугольного треугольника a, b. Вычислить длину гипотенузы c.

```
Program treug;
var a, b ,c: real;
begin
write ('a='); read (a);
write ('b='); read (b);
c:=sqrt (a*a+b*b);
writeln ('c=', c:6:2);
end.
```

Ветвление
Задача. Сравнить значения двух переменных X и Y.

```
Program Sorting;
var x, y, c: real;
begin
readln (x,y);
if x>y then writeln (‘x>y’) else writeln (‘x<y’);
end.
```

Цикл
Задача. Составить программу, которая вычисляет и печатает площади 20 квадратов со сторонами от 1 до 20.

```
Program two;
var
x,s: real;
Begin
x:=1;
writeln (‘сторона      площадь’);   {печать шапки таблицы}
while x<=20  do    {пока выполняется условие выполняй}
begin
s:=x*x;                          {тело цикла}
writeln (x, ‘              ‘, s);
```

```
x:=x+1;          {изменение параметра цикла с шагом 1}
end;
end.
```