Основы алгоритмизации и программирование

1. Этапы решения задач на ЭВМ
2. Понятие алгоритма
3. Свойства алгоритмов
4. Представление алгоритмов
5. Графический способ описания алгоритмов
6. Структурный подход к разработке алгоритмов

Этапы решения задач на ЭВМ

1. Постановка задачи (При постановке задачи выясняется конечная цель и вырабатывается общий подход к решению задачи. Выясняется сколько решений имеет задача и имеет ли их вообще. Изучаются общие свойства рассматриваемого физического явления или объекта, анализируются возможности данной системы программирования.)

2. Формализация (математическая постановка) (На этом этапе все объекты задачи описываются на языке математики, выбирается форма хранения данных, составляются все необходимые формулы.)

3. Выбор (или разработка) метода решения (Выбор существующего или разработка нового метода решения (очень важен и, в то же время личностный этап)).
4. Разработка алгоритма (На этом этапе метод решения записывается применительно к данной задаче на одном из алгоритмических языков (чаще на графическом)).
5. Составление программы (Переводим решение задачи на язык, понятный машине.)
6. Отладка программы
7. Вычисление и обработка результатов.

Понятие алгоритма

Алгоритм (*лат. algorithmi*)

В IX веке Абу Джафар ибн Муса аль-Хорезми сформулировал правила выполнения арифметических действий над числами в десятичной системе счисления.

Исполнитель алгоритма – это тот объект или субъект, для управления которым составлен алгоритм.

Вся совокупность команд, которые данный исполнитель умеет выполнять, называется *системой команд исполнителя* (СКИ).

Алгоритм - понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленной цели.

Свойства алгоритмов

1. ***Дискретность***
2. ***Массовость***
3. ***Результативность***
4. ***Понятность***
5. ***Определенность***
    - однозначность
    - детерминированность

Определение алгоритма в интуитивном смысле

Алгоритм – это правило или правила, сформулированные на некотором языке и определяющие пошаговый процесс обработки допустимых исходных данных в искомые результаты.

Каждый алгоритм связан с двумя языками:

- алгоритмический язык;
- язык операндов.

Способы представления алгоритмов

1. Словесный – описание алгоритмов на естественном языке;
2. Структурно-стилизованный способ (псевдокоды);
3. Графический;
4. Программный.

Графический способ описания алгоритмов

Блок-схема – это графическая интерпретация алгоритма, представляющая набор геометрических фигур, каждая из которых изображает какую-либо операцию или действие.

Форма символов и правила составления схем алгоритмов установлены государственными стандартами: ГОСТ 19.701 – 90 «Схемы алгоритмов, программ, данных и систем».

При построении алгоритмов на языке блок-схем руководствуются правилами:

1) Блок-схема строится сверху вниз.

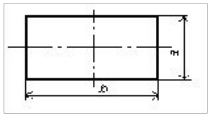
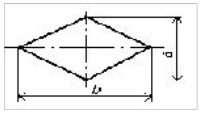
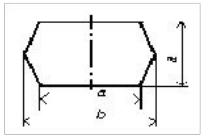
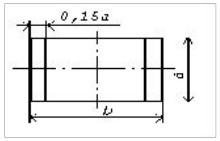
2) В любой блок-схеме имеется только один элемент, соответствующий началу алгоритма, и один элемент, соответствующий концу алгоритма.

3) Должен быть хотя бы один путь из начала блок-схемы к любому элементу.

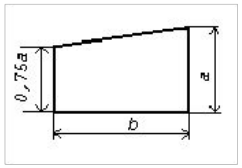
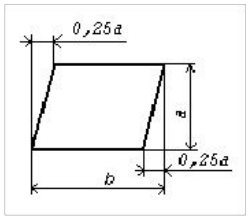
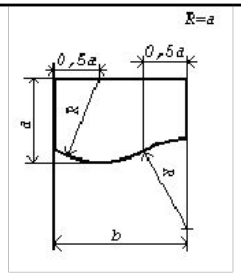
4) Должен быть хотя бы один путь от каждого элемента блок-схемы в конец блок-схемы.

Составление алгоритмов графическим способом подчиняется двум ГОСТам:
ГОСТ 19.002-80, соответствует международному стандарту ИСО 2636-73. Регламентирует правила составления блок-схем.
ГОСТ 19.003-80, соответствует международному стандарту ИСО 1028-73. Регламентирует использование графических примитивов.

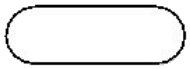
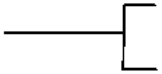
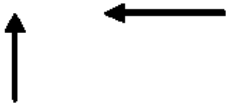
Основные символы схем алгоритмов

Наименование	Обозначение и размеры в мм	Функция
Процесс		Выполнение операций или группы операций, в результате которых изменяется значение, форма представления или расположение данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Модификация		Выполнение операций, меняющих команды или группу команд, изменяющих программу
Предопределенный процесс		Использование ранее созданных и отдельно описанных алгоритмов или программ

Основные символы схем алгоритмов

Наименование	Обозначение и размеры в мм	Функция
Ручной ввод		Ввод данных вручную при помощи неавтономных устройств с клавиатурой, набором переключателей, кнопок
Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)
Документ		Ввод-вывод данных, носителем которых служит бумага

Основные символы схем алгоритмов

Наименование	Обозначение и размеры в мм	Функция
Пуск - останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
Комментарий		Связь между элементами схемы и пояснением
Соединитель		Указание связи между прерванными линиями потока, связывающими символы
Линии потока		Указание последовательности связей между символами
Размер “a” должен выбираться из ряда 10, 15, 20 мм. Допускается увеличивать размер “a” на число, кратное 5. Размер “b” равен 1,5a.		

Структурный подход к разработке алгоритмов

Структурная алгоритмизация основывается на *двух принципах*:

1. последовательная детализация «сверху-вниз»;
2. ограниченность базового набора структур для построения алгоритмов любой степени сложности.

Требования структурного программирования:

1. программа должна составляться мелкими шагами, таким образом, сложная задача разбивается на достаточно простые, легко воспринимаемые части;
2. логика программы должна опираться на минимальное число достаточно простых базовых конструкций.

Основные структуры алгоритмов – это ограниченный набор стандартных способов соединения этапов (блоков) алгоритма для выполнения типичных последовательностей действий.

Базовый набор структурной алгоритмизации:

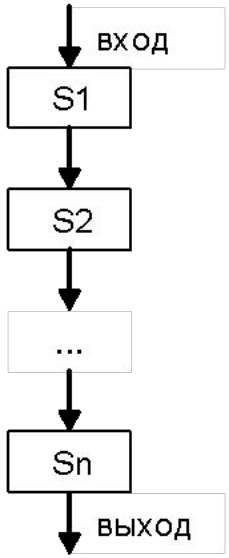
линейные,

разветвляющиеся,

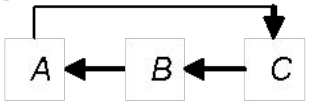
циклические структуры.

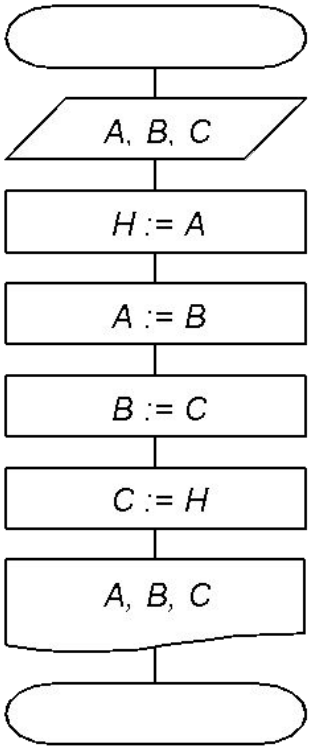
Линейные алгоритмы

Линейные алгоритмы – последовательность блоков, каждый из которых имеет по одному входу и одному выходу, и выполняется в программе один раз.



Пример 1. Записать алгоритм циклического перемещения влево значений между переменными *A, B, C.* Схема циклического перемещения:





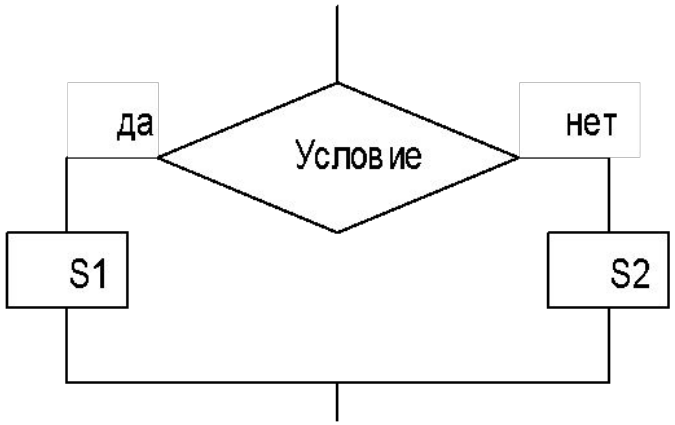
Алгоритм разветвленной структуры

Алгоритм разветвляющегося вычислительного процесса – алгоритм, в котором в зависимости от значений некоторого признака производится выбор одного из нескольких направлений, называемых ветвями.

В основе организации разветвления лежит проверка логического условия, которое может быть истинно или ложно. Частный вид логического условия – это операции типа =, ≠, >, <, ≥, ≤. Допускается также объединение нескольких условий в одно при помощи логических операций и, или, не и др.

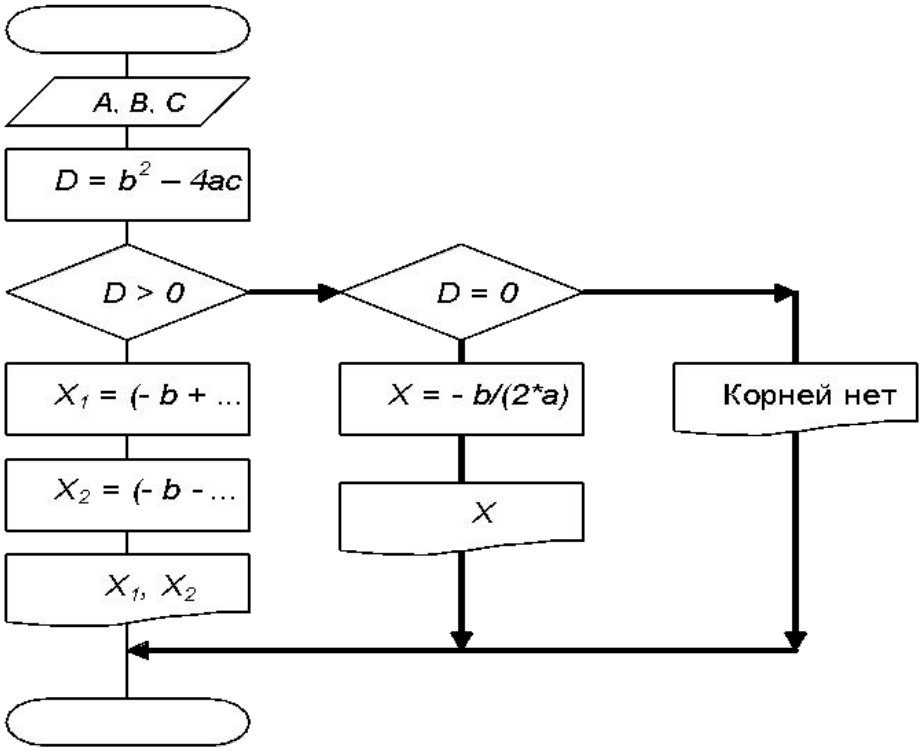
Алгоритм разветвляющейся структуры принято называть развилкой. Частный случай разветвления, когда одна ветвь не содержит никаких действий, называют *обходом* или *неполной развилкой*.

Схема базовой структуры ветвления:



Пример 2: Вычислить корни квадратного уравнения: $ax^2 + bx + c = 0$.

Решение: Рассмотрим простейший случай: коэффициенты уравнения a, b и c представляют собой вещественные числа и отличны от нуля.



Алгоритм циклической структуры

Алгоритм циклического вычислительного процесса включает в себя многократно повторяющиеся участки вычислений для различных значений данных.

Тело цикла – та последовательность действий, которая выполняется многократно.

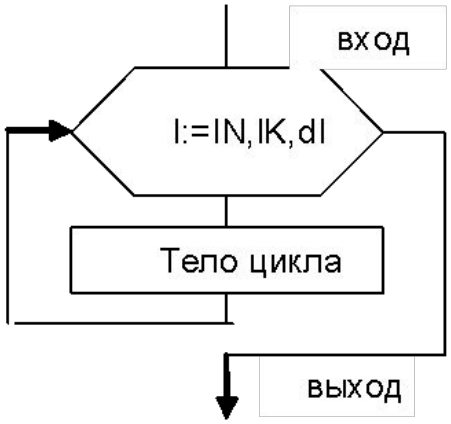
Для организации цикла необходимо выполнить следующие действия:

- Перед началом цикла задать начальные значения параметров (переменных, используемых в логическом выражении, отвечающем за продолжение или завершение цикла;
- Внутри цикла изменять переменную (или переменные), которая сменит значение логического выражения, за счет которого продолжается цикл, на противоположное;
- Вычислять логическое выражение – проверять условие продолжения или окончания цикла;
- Управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из цикла в противном случае.

Регулярный цикл (цикл с пересчетом) задается в тех случаях, когда число повторений заранее известно или легко может быть вычислено.

Для организации такого цикла необходимо задать начальное значение параметра – IN, конечное значения значение параметра – IK, шаг изменения параметра – dI.

Схема регулярного цикла:



Циклы итеративного типа

Цикл с предусловием (цикл «Пока») отличается тем, что проверка условия проводится до выполнения тела цикла «до тех пор, пока логическое выражение истинно выполнять операторы тела цикла».

Цикл с постусловием (цикл «До») отличается тем, что проверка условия выхода из цикла осуществляется после выполнения тела цикла. Тело цикла будет выполняться до тех пор, пока логическое выражение ложно.

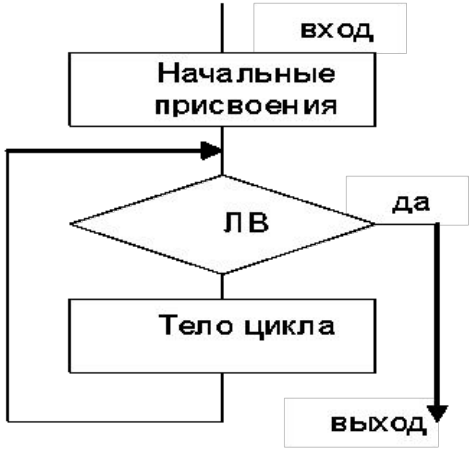


Схема цикла с предусловием

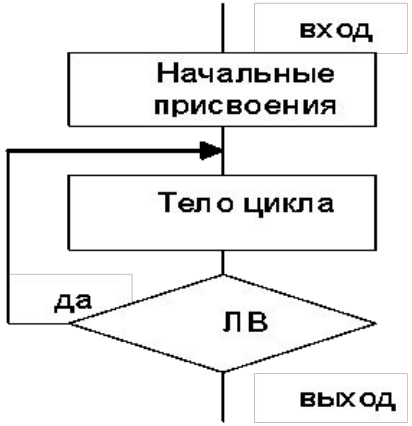
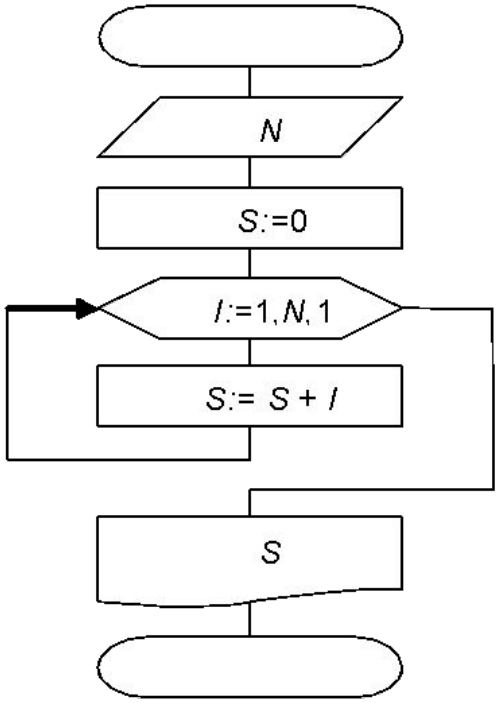


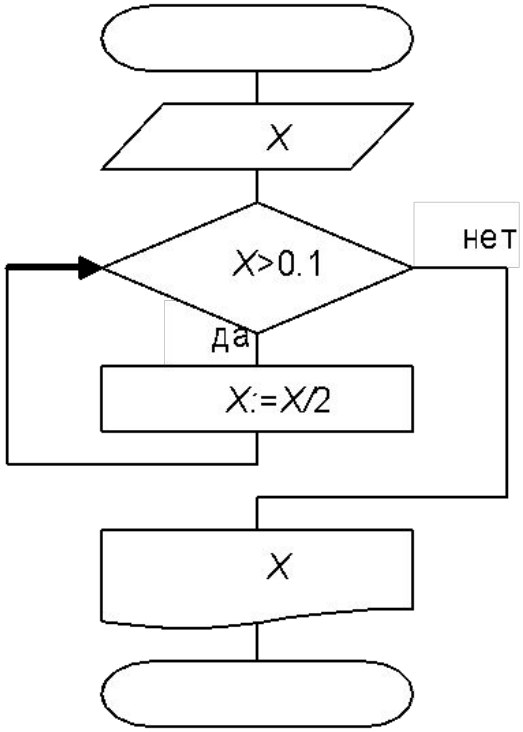
Схема цикла с постусловием

Пример 3: Вычисление суммы всех целых чисел от 1 до *N*.
Решение: для решения задачи используем цикл с параметром, где *I* – параметр цикла (счетчик целых чисел), *N* – конечное значение параметра (количество целых чисел), *S* – сумма всех целых чисел от 1 до *N*. Схема алгоритма:



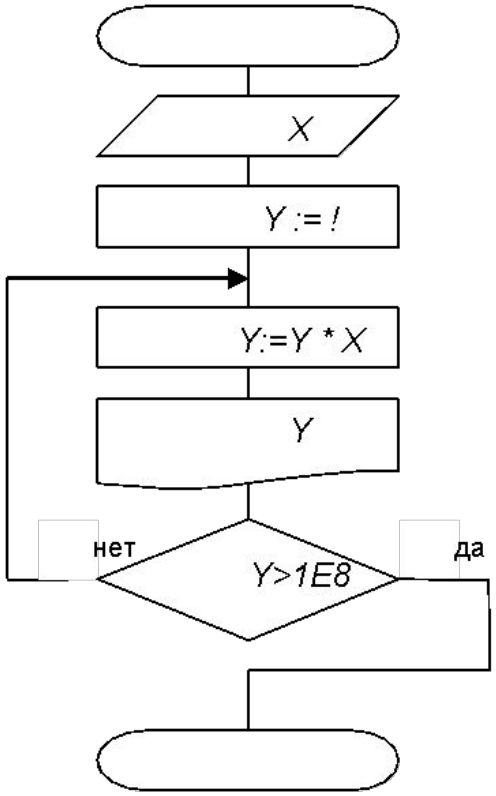
Пример 4: Дано X. Надо делить его пополам до тех пор, пока X будет больше 0,1.

Решение: организуем цикл с предусловием. Схема алгоритма:



Пример 5: Дано X>1. вычислить и вывести степени X до тех пор, пока вычисленное значение станет больше 10^8.

Решение: организуем цикл с постусловием. Схема алгоритма:



Программная реализация алгоритмов линейной структуры

Программа на языке Паскаль состоит из заголовка, раздела описаний и раздела операторов.

Заголовок программы состоит из ключевого слова Program и собственно имени программы, например: Program prim;. Заголовок является необязательной частью программы.

Описание (объявление данных) содержит упоминание всех объектов, используемых в программе и включает в себя:

- раздел подключаемых библиотек (модулей) определяется служебным словом USES и содержит имена подключаемых модулей: uses CRT, Graph;
- раздел описания меток: любой оператор в программе может быть помечен меткой. Имя метки задается по правилам образования идентификаторов Турбо Паскаль: label 3, 471, 29, Quit;
- раздел описания констант позволяет использовать имена как синонимы констант, их необходимо определить в разделе описания констант:

  const K= 1024; MAX= 16384;
- раздел описания типов;
- раздел описания переменных; необходимо указать все переменные, используемые в программе, и определить их тип: var P,Q,R: Integer; A,B: Char; F1,F2: Boolean;
- раздел описания процедур и функций.

Операторы (исполняемая часть) представляет собой составной оператор, который содержится между служебными словами begin.....end. Операторы отделяются друг от друга символом ”;”. Текст программы заканчивается символом ”точка”.

Алфавит языка Паскаль

- Латинские прописные и строчные буквы A – Z, a – z; символ _ «подчерк» (код ASCII 95) используются для формирования идентификаторов и служебных слов;
- Арабские цифры 0 – 9 – для записи чисел и идентификаторов.
- Специальные символы: математические (+| - | * | / | = | > | < | (|) |; пунктуации | . | , | : | ; |; прочие | [|] | - для обозначения массивов и множеств, | { | } | - для записи комментариев, | _ | - для разделения лексем, | ' | - апостроф для записи констант символьного и текстового типа, | $ | # | @ | ^ |.

Лексическая структура языка Паскаль

- Ключевые (служебные, зарезервированные) слова. Всего 51 слово, изображаются белым цветом.
- Идентификаторы (изображаются желтым цветом) могут быть двух разновидностей:

имена присваиваются какой-либо переменной, константе, типу, метке, процедуре или функции;

стандартные идентификаторы, которые являются именами встроенных в язык процедур и функций.

- Знаки операций: |= | < | > | + | - | * | <= | >=|.
- Изображения – эта группа лексем обозначает:
  - Десятичные числа, с фиксированной точкой: <Вещ_фикс>::=<целое>.< целое> или с плавающей точкой): <Вещ_эксп>::=<Вещ_фикс>E<порядок>, где <порядок>::=[+-] <целое>. Например: 7,4E-2 ⇒ 7.4 · 10-2 ⇒ 0.074.
  - Строки – последовательность любых символов из расширенного набора ASCII, заключенная в апострофы;
  - Комментарии (изображаются серым цветом) – любая комбинация произвольных символов, заключенная либо в фигурные скобки { }, либо в комбинированные {* *}.

Типы данных

Классификация предопределенных типов языка Паскаль

Группа	Подгруппа	Название	Идентификатор
Простой	Порядковый (ординальный)	Короткий целый	ShortInt
		Байтовый	Byte
		Слово	Word
		Целый	Integer
		Длинный целый	LongInt
		Символьный	Char
		Булев	Boolean
	Вещественный	Вещественный	Real
		С одинарной точностью	Single
		С двойной точностью	Double
		С повышенной точностью	Extended
		Сложный	Comp
Строковый			String
Структурный		Массив	Array
		Множество	Set
		Файл	File
		Запись	Record
Ссылочный			Pointer
Процедурный		Процедура	Procedure
		Функция	Function
Объектный			Object

Описание констант

Константы используются в программе для задания значений, которые не изменяются в процессе выполнения действий.

Предложение описания констант имеет вид:

Const c1 = value; c2 = value;

где c1, c2 – имена констант;

value – определяет значение константы.

Например:

Const c1 = -46.175; c2 = 1/c1; c3 = ABS(c1);

Описание переменных.

Переменные используются для записи значений, изменяющихся в программе.

Все переменные, используемые в программе, должны быть перечислены в разделе описания переменных:

Var v1, v2, … : type_id;

где v1, v2,… - список переменных;

type_id задает тип переменных из данного списка.

Например:

Var a, b, c: integer; x1, x2: real;

Операции и выражения

Выражения – это конструкции, которые могут включать в себя константы, переменные, стандартные функции, пользовательские функции и числа, соединенные между собой знаками операций и парами круглых скобок. Выражения состоят из операндов и операций, записываются в одну строку и всегда имеют конечное значение определенного типа.

Операции по количеству операндов делятся на

унарные (присвоение знака числу [+ -];

бинарные (умножение [*], деление [/], деление нацело [div], остаток от деления [mod], сложение [+], вычитание [-].

Например: C := A mod B, при A := 13 и B := 4 C := 1;

C := A div B, при A := 13 и B := 4 C := 3.

Стандартные функции

Результат работы функции возвращается в виде значения этой функции. В выражении функция вычисляется в первую очередь, если нет скобок.

Abs(X) – вычисляет абсолютное значение X;

Exp(X) – основание натурального логарифма возводит в степень X;

Ln(X) – вычисляет натуральный логарифм X;

Sqr(X) – X возводит в квадрат;

Sqrt(X) – вычисляет квадратный корень из X;

Sin(X), Cos(X), Arctan(X) – тригонометрические функции синус, косинус и арктангенс (аргумент задается в радианах);

Trunc(X) – определяет целую часть числа X, тип результат Longint;

Round(X) – округляет число X до целого;

Frac(X) – определяет дробную часть аргумента;

Int(X) – определяет дробную часть аргумента, тип результат Real;

Операторы

Это единицы действия языка (предложения, утверждения, инструкции).

Выполняемые операторы производят вычисления и управляют процессом вычислений.

Невыполняемые операторы содержат сведения о структуре и организации данных, их свойствах и размещении данных в памяти.

Составной оператор – это последовательность произвольных операторов программы, заключенная в операторные скобки *Begin ... End*.

Пустой оператор не содержит никаких действий, просто в программу добавляется лишняя точка с запятой.

Оператор присваивания вызывает выполнение выражений и присваивание значения имени результата: <имя переменной> := <выражение>.

Например:

C := A/B;

D := 54 *S +(21 +n)/g);

P := P*i.

Операторы ввода-вывода.

Для **ввода данных** используются следующие операторы обращения к встроенной стандартной процедуре ввода данных:

Read (A, B, C); где A, B, C – имена переменных, значения которых подлежат вводу для запоминания в оперативной памяти.

Readln (A, B, C); после окончания ввода курсор перемещается к началу новой строки.

Readln; означает ожидание нажатия клавиши <Enter>.

Для **вывода данных** на экран монитора используется оператор обращения к стандартной процедуре вывода данных:

Write(список имен);

Например:

Write(‘a=’, a:7:3, ‘_b=’, b:6:3)

В приведенном операторе ввода

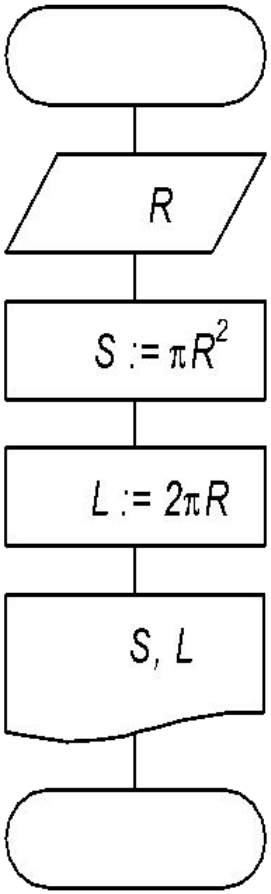
при a = 3,14744 и b = -3,4 на экране выведется

a = __3.147 b = -3.400.

Writeln(список имен); - после вывода курсор переводится к началу новой строки.

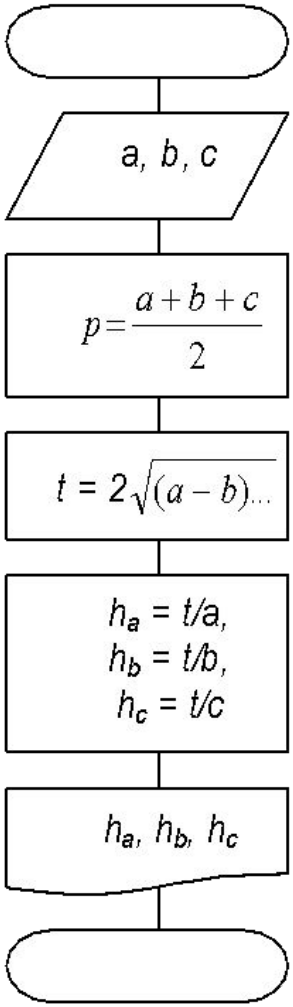
Writeln; - означает пропуск одной строки и переход к началу новой строки.

Пример 1. Вычислить площадь круга и длину окружности по заданному радиусу.




```
program circle;
const  pi := 3.14159;
var r, s, l : real;
begin
        writeln ('введите радиус'); readln (r);
        s:= pi*r*r; l:= 2*pi*r;
        writeln ('площадь круга =', s:8:4);
        writeln ('длина окружности =', l:8:4);
end.
```

Пример 2. Вычислить высоты сторон треугольника, если известны длины сторон треугольника. Для вычисления использовать формулу Герона.




```
program primer2;
var a, b, c, p, t, ha, hb, hc: real;
begin
        writeln ('введите длины сторон треугольника a, b,
        c'); readln (a, b, c);
        p:= (a + b + c)/2; p:= 2*sqrt(p * (p – a) * (p – b) * (p – c));
        ha:= t/a; hb:= t/b; hc:= t/c;
        writeln ('ha =', ha:8:4);
        writeln ('hb =', hb:8:4);
        writeln ('hc =', hc:8:4);
end.
```

Оператор условного перехода

if <*условие*> then <*оператор 1*> else <*оператор 2*>;

где if, then, else – зарезервированные слова (если, то, иначе),

<условие> - логическое выражение, в зависимости от которого выбирается одна из двух альтернативных ветвей алгоритма,

<*оператор 1*> и <*оператор 2*> - любые операторы языка Паскаль. Если значение условия истинно (TRUE), то будет выполняться <*оператор 1*>, записанный после ключевого слова then. В противном случае будет выполнен <*оператор 2*>, следующий за словом else, при этом <*оператор 1*> пропускается.

Логические операции

NOT – отрицание;

AND – конъюнкция;

OR - дизъюнкция,

XOR - строгая дизъюнкция.

Логические операции применимы к операндам целого и логического типов. Если операнды целого типа, то результат логической операции есть тоже целое число.

Логические операции над логическими данными дают результат логического типа. Правила выполнения логических операций отражены в таблице истинности.

Таблица истинности

A	*B*	*not A*	*A and B*	*A* or *B*	*A* xor *B*
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	0	1
1	1	0	1	1	0

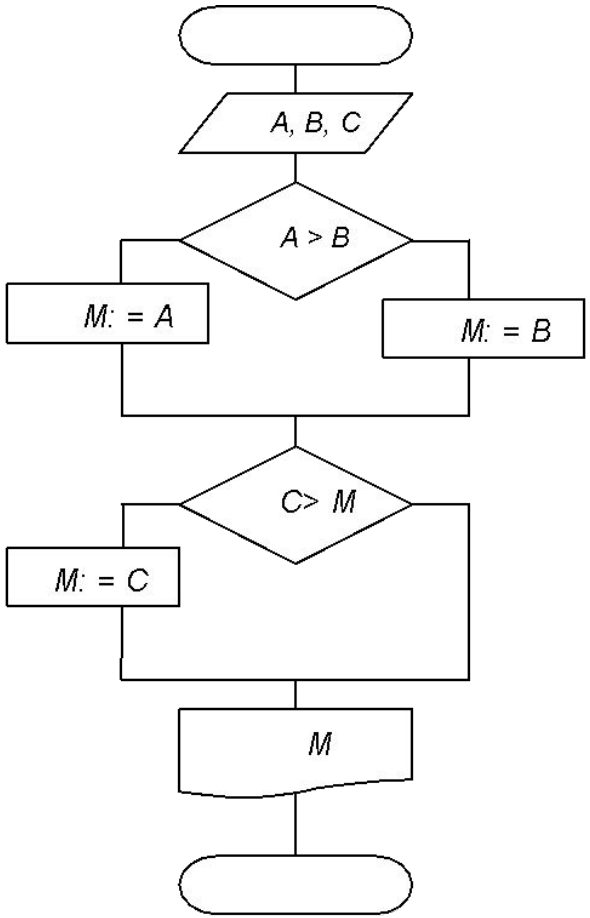
Логические операции имеют более высокий приоритет, чем операции отношения. В связи с этим, в сложных логических выражениях обычно необходимо расставлять скобки.

Например, b и c имеют тип integer, то выражение *a = b and c < d* вызовет сообщение об ошибке, т.к. сначала выполнится операция *b and c*. Правильным будет выражение: (*a = b*) *and* (*c < d*).

Пример 3. Из трёх предложенных чисел A, B и C выбрать и вывести максимальное.

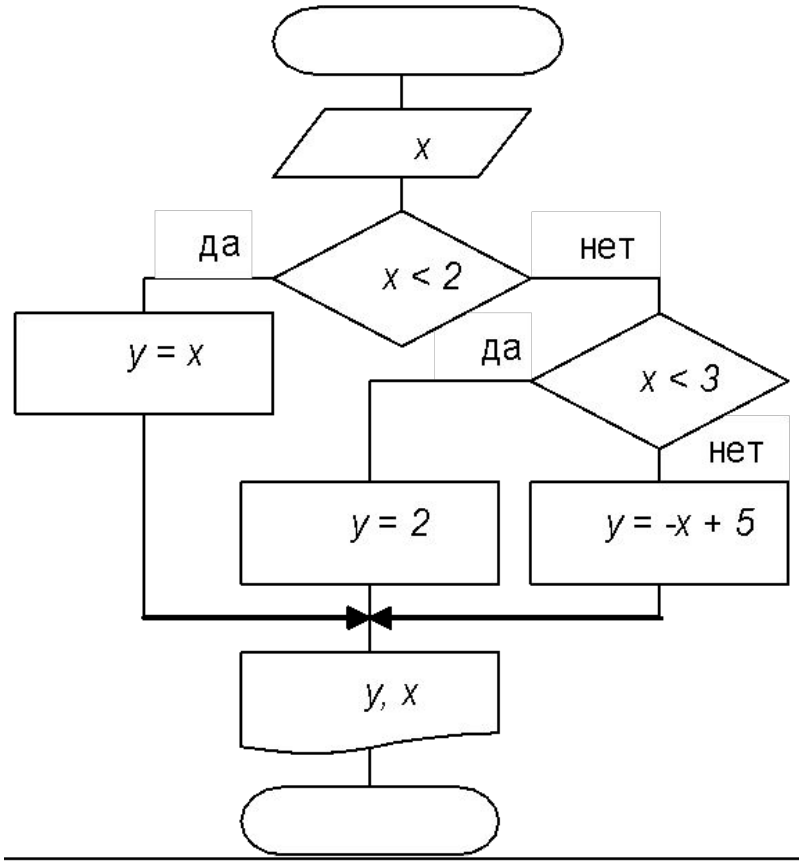
Решение: Сначала сравним между собой A и B, наибольшее значение сохраним во вспомогательную переменную M. Затем, если окажется, что C>M, то заменим значение M; в противном случае никаких действий предпринимать не нужно.

```
program primer1;
var a, b, c, m: real;
begin
writeln (‘введите числа  a, b, c’); readln (a, b, c);
if a > b then m:= a else m:= b;
if c > m then m:= c;
writeln (‘максимальное из a, b, c’, m:8:4);
readln
end.
```


A, B, C
A > B
M: = A
M: = B
C> M
M: = C
M

Пример 4. Вычислить значение функции

$$y = \begin{cases} x, \text{если } x \le 2 \\ 2, \text{если} 2 \le x < 3 \\ -x + 5, \text{если } x \ge 3 \end{cases}$$



Программный код:

```
program primer3;
var x, y: real;
begin
  writeln ('введите x'); readln (x);
if x < 2 then y:=x
          else if x < 3 then  y:= 2
                        else y:= -x + 5;
  writeln ('y=', y, 'при x=' x);
  readln
end.
```

Арифметический оператор цикла (цикл с пересчетом, регулярный цикл)

for I := IN to IK do S; или for I := IN downto IK do S;

где I - параметр цикла, управляющая переменная, идентификатор;

IN, IK – выражения для определения начального и конечного значений параметра;

to определяет шаг изменения параметра цикла: dI = 1; downto определяет шаг изменения параметра цикла: dI = -1;

S - один оператор, простой или составной, тело цикла.

Правила формирования и выполнения цикла for:

- I, IN и IK могут быть любого ординального типа, в т.ч. целыми, логическими, перечисляемыми и диапазонными;
- I, IN и IK должны быть совместимого типа;
- Значение I не должно изменяться операторами в теле цикла;
- Анализ цикла производится до выполнения тела цикла, поэтому при IN > IK для to и при IN < IK для downto тело цикла не выполнится ни разу;
- После завершения цикла for параметр цикла имеет значение: после нормального завершения I = IK или при котором выполняется оператор goto;
- Значения IN и IK определяются один раз, в начале выполнения оператора for, и сохраняются во время выполнения оператора for;
- dI определяет для I целого типа приращения значения I, а для параметров перечисляемого типа – приращение номера значения I;
- для каждого I тело цикла выполняется один раз;
- после каждого выполнения тела цикла к параметру цикла I добавляется его приращение dI.

Оператор итерационного цикла с предусловием

while B(X) do S;

где B(X) - логическое выражение, условие завершения цикла; S - один оператор, простой или составной, тело цикла.

Оператор while используется, когда число повторений операторов тела цикла заранее неизвестно и определяется в процессе выполнения цикла. Операторы тела цикла при определенных условиях не должны выполняться ни разу.

Правила выполнения цикла while:

- в теле цикла S должны быть операторы для изменения значений операндов логического выражения B(X);
- Тело цикла выполняется только в случае, если результат выражения B(X) истинный (True), если он ложный тело цикла не выполняется и происходит выход из цикла.

оператор итерационного цикла с постусловием

repeat S

until B(X);

где B(X) - логическое выражение, условие завершения цикла; S - один оператор, простой или составной, тело цикла.

Оператор repeat используется, когда число повторений операторов тела цикла заранее неизвестно и определяется в процессе выполнения цикла. Операторы тела цикла должны выполниться хотя бы один раз.

Repeat и until B(X) определяют начало и условие завершение цикла.

После каждого выполнения тела цикла анализируется значение результата логического выражения B(X): если оно ложно (False), выполнение тела цикла повторяется, если истинно (True) – цикл завершается.