Т.В. Волобуева

ИНФОРМАТИКА

Основы программирования на языке Pascal

Учебное пособие

Воронеж 2019

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Воронежский государственный технический университет»

Т.В.ВОЛОБУЕВА

ИНФОРМАТИКА

Основы программирования на языке Pascal

Учебное пособие

Воронеж 2019

УДК 69.003:658.5.012.22
ББК 32.973
В68

Рецензенты:

Кафедра высшей математики и информационных технологий Воронежского государственного университета инженерных технологий;

Корелина Т.В., К.т.н

В68 **Волобуева, Т.В.**
Инфоорматика. Основы программирования на языке Pascal: методическое пособие / Т.В.Волобуева ; Воронеж. гос. технич. ун-т. – Воронеж, 2019. – 110 с

ISBN

Изложен теоретический материал по основам программирования на языке Pascal. Приведены примеры программ и задания для самостоятельной работы. Является продолжением пособия «Информатика. Основы алгоритмизации».. Рекомендуется к изучению после прочтения пособия «Информатика. Основы алгоритмизации».

Ил. 69 Табл. 5., Библиограф.: 8 названий

УДК 69.003:658.5.012.22
ББК 32.973

Печатается по решению учебно-методического совета

Воронежского государственного технического университета

ISBN

ВВЕДЕНИЕ

Паскаль (Pascal), разработан Н. Виртом (швейцарская высшая техническая школа, Цюрих) как средство для обучения технике и принципам программирования. Это хороший язык для того, чтобы научиться реализации алгоритмов в коде (на языке программирования), так как является компромиссом между простотой реализации и точностью написания кода, а также способствует формированию у начинающего программиста алгоритмического и логического мышления при выполнении задач.

Программирование – прежде всего это алгоритмы, и начинать надо с них. Но чтобы научиться практическому применению алгоритмов, нужен инструмент для работы с ними, язык программирования. Программирование подразумевает строгость рассуждений и преобразований, логику, то, что принято считать качествами математического мышления. И первый язык программирования должен помогать формированию такого мышления.

Язык паскаль очень привлекателен тем, что дает возможность писать программу почти как на обычном английском языке. Т.е. синтаксис языка Паскаль – не представляет особых проблем. Привычные английские слова в Паскале, интуитивно понятный код языка, строгая структуризация и типизация учат по-настоящему правильному программированию

Программу нужно писать так, чтобы ее:

1. Можно было легко читать/
2. Ее можно было легко использовать.

Важной ориентацией при написании программы должно быть то, что ваш программный продукт должен удовлетворить потребности людей, которые будут его читать, и использовать.

Язык Паскаль облегчает программисту задачу написания понятных программ т.к. обладает хорошими средствами для их проектирования. В этом пособии критерием качества программы будем считать не минимальное количество операторов в программе и минимальное время ее исполнения, а понятность ее другим людям

Паскаль удобен для обучения основам программирования, воспитания правильных привычек и стиля построения программ. При изложении материала в этом пособии использованы только те конструкции, которые применяются и в других процедурных языках программирования.

Излагаемый в данном пособии материал ограничен только теми темами, которые были изложены в методическом пособии «Информатика. Основы алгоритмизации» и является логическим продолжением этого пособия.

Пособие направлено на формирование начальных навыков программирования. В нем рассмотрены операторы и структуры языка, используемые для описания основных типов вычислительных процессов.

Приведены примеры решения задач и задания для самостоятельной подготовки. Пособие адаптировано для студентов первокурсников.

Классификационные признаки паскаля

В современном мире существует огромное количество языков программирования. Ориентируются они на различные области применения. Возникает потребность классификации языков программирования. Таких попыток было много Паскаль, например, относят к классу процедурных языков программирования. Существуют также логические, объектно-ориентированные и другие группы языков. Такое деление основано на использовании в каждой группе различных принципов программирования.

В процедурных языках программирования программа состоит из последовательности действий (операторов), которые явно выполняют какие то модификации над данными. Таким образом, программа позволяют представить решение задачи как точный алгоритм.

Среди процедурных языков выделяются в свою очередь структурные и операционные языки.

В структурных языках одним оператором записываются целые алгоритмические структуры (ветвления, циклы и т.д.). В операционных языках для этого используются несколько операций. Пример структурных языков программирования: Паскаль, Си, Ада, ПЛ/1, Фортран, Бейсик, Фокал, Модула. Паскаль также обладает и возможностями операционного языка.

Объектно-ориентированная модель была добавлена в Turbo Pascal. Object Pascal (диалектом которого является Delphi), является функциональным расширением языка Turbo Pascal.

Этапы написания программы

Условно процесс написания программы можно разбить на три этапа:

1. Постановка задачи, построение математической модели, поиск или разработка алгоритма решения, определения порядка ввода исходных данных и вывода результатов решения задачи.
2. Написание исходного кода (текста программы на выбранном языке программирования).
3. Отладка и тестирование программы.

Отладка программы – это процесс устранения ошибок из текста программы. Все возможные ошибки в программе делятся на синтаксические и логические. При наличии синтаксических ошибок (ошибок в написании операторов языка программирования) программа не запускается. Логические ошибки — это ошибки, при которых программа работает неправильно, т.е. выдает не те результаты, которые должны быть получены при заданных начальных условиях. Иногда для исправления

логических ошибок приходится переписывать отдельные участки программы, иногда перерабатывать весь алгоритм. Тестированием программы называется процесс выявления ошибок в ее работе, приводящим к неверным результатам. Процессы отладки и тестирования сопровождаются многократным запуском программы на выполнение с различными вариантами исходных данных.

Реализации первого этапа в процессе написании программы (пункт первый из списка выше) призвано помочь методическое пособие «Информатика. Основы алгоритмизации». В данном пособии займемся реализацией этапов 2, 3 из указанного списка. Т.е. первоначально – алгоритм решения задачи, затем программа, реализующая его исполнение на ЭВМ, а не наоборот.

Задача любой программы – состоит в обработке данных. Данные в Паскале делятся на константы (не меняют своего значения в ходе выполнения программы) и переменные (могут менять свое значение). Данные в Паскале характеризуются именем, типом и значением. Умолчаний в Паскале нет. Все величины (данные) алгоритма должны быть описаны.

Структура программа на языке Паскаль

В программе записывается последовательность действий, выполняемая над данными для достижения поставленной цели. Программа – это способ записи алгоритма. При подготовленном алгоритме решения задачи – написание программы носит механический характер. Требуется только заменить каждое действие (шаг или структуру алгоритма) соответствующей командой языка программирования.

Программа на Паскале состоит из:

1. Заголовка программы.
2. Программного блока.
   2.1. Раздела описаний
   2.2. Раздела операторов.

Заголовок – состоит из зарезервированного слова program и имени программы. Заголовок может отсутствовать. Но, поскольку он несет смысловую нагрузку, для более быстрого поиска нужной программы рекомендуется его использовать при написании программы.
Например,

program poisk_maksimuma;

В конце любого предложения Паскаля следует ставить точку с запятой (;) – этот символ является разделителем предложений Паскаля.

Программный блок состоит из двух разделов.

В разделе описаний должны быть описаны все имена (идентификаторы), встречающиеся в программе. При описании следует придерживаться в такой последовательности. Описываются:

1. Подключаемые библиотечные модули (**uses**);
2. Описываются метки (**label**);
3. Описываются константы (**const**);
4. Типы данных (**type**);
5. Переменные (**var**);
6. Процедуры и функции (**procedure**, **function**).

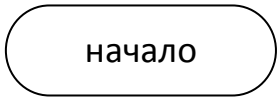
В скобках приведены зарезервированные слова, с которых начинается соответствующее описание.

При описании констант и переменных после соответствующего зарезервированного слова следует список величин одного типа (имена величин, разделенные запятой), и после двоеточия – указывается тип данных.

Любой элемент приведенного списка может отсутствовать

Имена величин (идентификаторы) не могут быть зарезервированными словами, могут быть любой длины, начинаться должны с буквы, не могут содержать специальных символов (пробелов, точек, звездочек и т.д.). Рекомендуется использовать подчеркивание в именах. Рекомендуется выбирать мнемонические имена, т.е. имена несущие смысловую нагрузку. Это удобно и человеку, пишущему программу, а главное тому, кто будет ее читать (использовать)

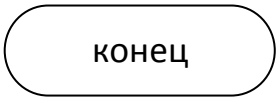
Второй раздел программного блока – раздел операторов, отделяется от раздела описаний словом begin. **Begin** – это открывающая операторная скобка и после нее не следует ставить никаких знаков препинания. На блок схеме это зарезервированное слово соответствует блоку



Далее следуют исполняемые действия (операторы) над объектами, описанными в разделе описаний. Разделяются они точкой с запятой

Заканчивается раздел операторов зарезервированным словом **end.** Это закрывающая операторная скобка.

На блок схеме она соответствует блоку



В конце программы следует ставить точку.

Синтаксис языка

Язык программирования – средство общения между человеком и компьютером. В любой знаковой системе (в том числе и языках программирования) синтаксис это правила построения сообщений в этой системе. Каждый язык программирования имеет строго определенную

грамматику, называемую синтаксисом языка. Если предложение языка программирования (оператор) не соответствует синтаксису, то оно не имеет смысла. Синтаксически правильный оператор языка программирования всегда имеет однозначную трактовку.

Типы данных Паскаля

При решении задач происходит обработка информации различного свойства (дробные числа, целые числа, слова и.т.д). Указание типа данных величины используют для описания множества допустимых значений величины, и совокупности операций в которых может участвовать эта величина. Каждый тип имеет зарезервированное слово для описаний. Все типы данных можно разделить на две группы:

– скалярные (простые типы);

– структурированные (составные).

Скалярные типы в свою очередь делятся на:

– стандартные типы (предлагаются разработчиками паскаля);

– пользовательские типы (разрабатываются программистами).

В этом пособии рассматриваются только простые стандартные типы данных

Целочисленные типы данных паскаля

Целочисленные типы паскаля представлены в табл. 1.

Таблица 1. Целочисленные типы паскаля

Тип (зарезервированное слово для описания)	Диапазон значений	Требуемая память (байт)
Byte	0…255	1
ShortInt	-128…127	1
Integer	-32768…3276	2
word	0…655357	2
Longint	-2147483648… 2147483647	4

Например, если введенное значение величины, описанное в программе как Byte, окажется больше допустимого диапазона значений от 0 до 255, то Вы получите ошибку ввода. Если величина, описанная как byte, в процессе выполнения алгоритма получит значение, выходящее за пределы допустимого диапазона от 0 до 255, то будет выведен мусор (в Паскале не контролируется возможное переполнение), т.е. опять получим ошибку вычисления. Поэтому описывая величины вашего алгоритма,

необходимо продумать какого типа будут эти данных, т.к. при выборе типа выделяется и соответствующая память под величину

Вещественные типы данных паскаля

Вещественные числа в паскале могут быть представлены в двух формах:

– с фиксированной точкой. Например, 2.5.

– с плавающей точкой mE±p. Здесь m – мантисса (целое или дробное число), E- означает 10 в степени, p – порядок (целое число). Например 25E-1. Это означает $25 * 10^{-1}$. Т.е. это тоже число 2.5. Форму записи с плавающей точкой удобно использовать при работе с очень большими или очень маленькими величинами. Вещественные типы данных паскаля представлены в табл. 2.

Таблица 2. Вещественные типы данных паскаля

Тип (зарезервированное слово для описания)	Диапазон значений	Требуемая память (байт)
Real	2.9E-39 … 1.7E38	6
Single	1.5E-45 … 3.4E38	4
Double	5.0E-324 ...1.7E308	8
Extended	3.4E-4932 … 1.1E493	10
comp	-9.2E63 … (9.2E63)-1	8

Символьный тип данных

Описывается зарезервированным словом char. Множество допустимых значений – множество значений кодовой таблицы ПК. Для размещения в памяти величины типа char требуется 1 байт.

Логический тип данных

Описывается зарезервированным словом Boolean. Величина этого типа может принять два значения true(истина) или false (ложь). Для размещения в памяти величины этого типа требуется 1 байт

Операции Паскаля

Все операции паскаля можно разделить на

1. Арифметические
2. Отношения
3. Логические
4. Строковые;
5. Разрядные, сдвиговые;

6. Операции над множествами; и т.д.

Подробнее ознакомимся с первыми тремя группами операций.

Арифметические операции могут быть бинарными (в операции участвуют два операнда) и унарными (в операции участвует один операнд). Арифметические операции паскаля приведены в табл. 3.

Таблица 3. Арифметические операции паскаля

Операция	Действие	Тип операндов	Тип результата
бинарные			
+	Сложение	Целый. Вещественный	Целый. Вещественный
-	Вычитание	Целый. Вещественный	Целый. Вещественный
*	Умножение	Целый. Вещественный	Целый. Вещественный
/	Деление	Целый. Вещественный	Вещественный
Div	Целочисленное деление	Целый	Целый
Mod	Остаток от деления	Целый	Целый
Операция	Действие	Тип операндов	Тип результата
Унарные			
+	Сохранение знака	Целый. Вещественный	Целый. Вещественный
-	Отрицание знака	Целый. Вещественный	Целый. Вещественный

Операции отношения выполняют сравнение двух операндов и определяют, истинна или ложь будет в результате выполнения операции. Сравниваемые величины могут принадлежать любому типу данных. Результат операции – всегда логического типа. Операции отношения представлены в табл. 4.

Таблица 4. Операции отношения паскаля

Операция	Название	Выражение на паскале
=	Равно	A=B
<>	Неравно	A<>B
>	Больше	A>B
<	Меньше	A<B
>=	Больше или равно	A>=B
<+	Меньше или равно	A<=B

Логические операции.

Логические выражения в результате вычисления принимают значения логического типа. Операндами логического выражения могут быть логические константы, переменные, отношения.

В паскале имеется 4 логические операции:

Not – логическое отрицание

And – логическое умножение

Or – логическое сложение

Xor – исключающее или

Таблица истинности приведена ниже (истина в таблице обозначена T. ложь – F). A и B – операнды, участвующие в операции.

Таблица 5. Типы результата выполнения логических операций паскаля

A	B	Not A	A and B	A or B	A xor B
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	E	T

Приоритет выполнения операций паскаля в порядке убывания. В одной группе, операции одного приоритета.

1. Вычисление функций;
2. Унарный минус, not;
3. Умножение, деление, div, mod, and;
4. Сложение, вычитание, or, xor;
5. Операции отношения.

Если в выражении присутствуют операции одного приоритета, то выполняются они с лева на право. Если приоритет операций в выражении нужно нарушить – используют круглые скобки.

Некоторые стандартные функции

Таблица 6. Некоторые стандартные функции паскаля

Обращение	Тип аргумента	Тип результата	Функция
Abs(x)	Целый. Вещественный	Целый. Вещественный	Модуль аргумента
Arctan(x0	Целый. Вещественный	Вещественный	Арктангенс
Cos(x0	Целый. Вещественный	Вещественный	Косинус
Exp(x0	Целый. Вещественный	Вещественный	e^x- экспонента
Sin(x)	Целый. Вещественный	Вещественный	Синус

Ln(x)	Целый. Вещественный	Вещественный	Натуральный логарифм
Sqrt(x)	Целый. Вещественный	Вещественный	Корень квадратный из x
Sqr(x)	Целый. Вещественный	Вещественный	Квадрат x
Frac(x)	Целый. Вещественный	Вещественный	Дробная часть x
Int(x)	Целый. Вещественный	Вещественный	Целая часть x

Ввод – вывод данных

Ввод данных – это передача информации от внешнего носителя в оперативную память для обработки.

Выполнение этой операции в паскале происходит путем обращения к стандартным процедурам **read** или **readln**

Запись процедуры ввода в общем виде:

read(список ввода);

В списке ввода могут быть только имена переменных. Они перечисляются через запятую. Значения этих переменных набираются с клавиатуры через пробел после запуска программы на выполнение. Значения переменных должны вводиться в строгом соответствии с синтаксисом языка (следует следить за типом вводимой величины). Если соответствие введенного значения и типа в описании вводимой величины нарушается, то произойдет ошибка ввода. Процедура Readln аналогична процедуре Read. Отличается только тем, что после ввода последнего в списке ввода значения курсор ввода перейдет в начало новой строки.

Например read(a,b,c);

Вывод – процесс обратный вводу. Т.е. данные после обработки из оперативной памяти передаются на внешний носитель.

Выполнение этой операции в паскале происходит путем обращения к стандартным процедурам **write** или **writeln**

Запись процедуры вывода в общем виде:

Write (список вывода);

В списке вывода могут быть - выражения допустимых типов данных, произвольный текст, заключенный в апострофы, имена переменных.

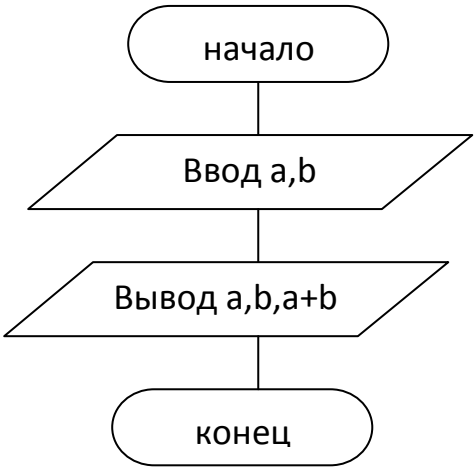
Например Write(’a=’,a,’b=’,b,’сумма равна-‘,a+b);

Процедура writeln – аналогична процедуре write. Отличие только в том, что после вывода последнего в списке вывода значения, курсов ввода перейдет на начало новой строки.

Пример 1. Вводятся два вещественных числа. Требуется вычислить их сумму. Вывести оба введенных числа и вычисленный результат.

Обозначения: a,b – введенные числа

Блок-схема алгоритма



Программа:

```
program summa;
var a,b:real;
begin
writeln ('введите два числа');
readln(a,b);
writeln ('a=',a,'    b=',b,'    сумма введенных
чисел =',a+b);
end.
```

Распечатка окна вывода

```
введите два числа
1 2
a=1    b=2    сумма введенных чисел =3
```

Комментарии к содержимому окна вывода.

Первая строка – результат работы оператора
writeln ('введите два числа'); т.е. на монитор в окно вывода распечатается текст, записанный в апострофах. Исходные данные задачи рекомендуется вводить в режиме диалога.

Вторая строка окна вывода – это числа 1 и 2 набранные пользователем через пробел, поскольку в данном случае требовалось вести любые два числа. После того как пользователь наберет эти два числа требуется нажать клавишу Enter (ввод) а оператор программы `readln(a,b);` – считает эти два числа в память, отведенную для переменные a,b соответственно

Третья строка окна вывода – результат работы оператора:

```
writeln ('a=',a,'    b=',b,'    сумма введенных чисел
=',s);
```

Здесь для того, чтобы при печати выводимые величины не сливались, использовались пробелы в тексте ' b=' содержит четыре пробела в выводимом тексте перед буквой b.

В выводимом тексте ' сумма введенных чисел =' перед словом сумма – четыре пробела.

Можно использовать при выводе форматный вывод.

Для вывода вещественных чисел в операторе WRITE необходимо указать формат представления такого числа, который имеет вид:

WRITE(имя_переменной:m:n);

где: m – целое число, указывающее общее количество позиций, отводимое под значение переменной, включая позицию под знак числа, точку и количество цифр дробной и целой части, n – целое число, определяющее количество цифр дробной части.

Выводимая информация выравнивается по левому краю выделяемого поля.

Вывод вещественных чисел выполняется по следующим правилам:

1. Если число имеет меньшую длину, чем m, то оно будет дополнено слева пробелами.
2. Если число имеет большую длину, чем m, то параметр m игнорируется и число будет выведено полностью.
3. Если дробная часть больше, чем параметр n, то число округляется. Округление не изменяет самого значения переменной, меняется только представление числа на мониторе.
4. Если параметр n не указан, то ни дробная часть числа, ни десятичная точка не выводятся. Вывод осуществляется в формате с плавающей точкой.

Для вывода целых чисел в операторе WRITE необходимо указать формат представления такого числа, который имеет вид:

WRITE(имя_переменной:k);

где: k– целое число, указывающее общее количество позиций, отводимое под значение переменной.

В целых числах отсутствует дробная часть, поэтому нет необходимости указывать в формате число позиций дробной части.

Программа с использованием форматного вывода в примере 1 имеет вид:

```
program summa;
var a,b:real;
begin
writeln ('введите два числа');
```

```
readln(a,b);
writeln ('a=',a:7:2,'    b=',b:7:2,'    сумма
введенных чисел =',a+b:7:2);
end.
```

Окно вывода:

```
введите два числа
1 2
a=   1.00    b=   2.00    сумма введенных чисел =
3.00
```

Операторы паскаля

Оператором называется предложение языка программирования, задающее полное описание некоторого действия, которое необходимо выполнить. Разделитель операторов в паскале точка с запятой. Операторы, не содержащие других операторов, называются простыми. В рамках этого пособия мы рассмотрим только один такой оператор – присваивания.

Структурные операторы представляют собой конструкции, построенные из других операторов по строго определенным правилам их можно разделить на три группы – составные, условные и повтора (цикла).

Оператор присваивания.

Оператор имеет структуру

Имя_переменной:=выражение;

Тип переменной в левой части этой конструкции должен соответствовать типу вычисленного выражения в правой ее части.

Оператор присваивания выполняется в таком порядке:

1. Вычисляется выражение, записанное справа от знака :=
2. Полученное в результате значение присевается переменной, записанной слева от знака := , т.е. переменная теперь будет хранить это значение.

Пример 1. Вычислить сумму двух введенных чисел. Вывести оба эти числа и вычисленный результат.

Обозначения:

a,b – введенные числа вещественного типа

s – сумма введенных чисел

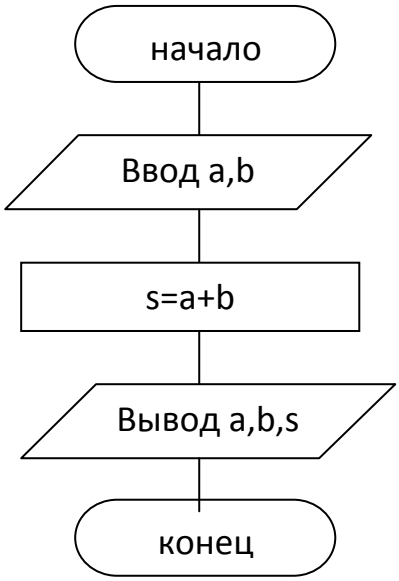


Рис.1. Блок-схема алгоритма решения задачи примера 1

Программа:

```
program summa;
var a,b,s:real;
begin
writeln ('введите два числа');
readln(a,b);
s:=a+b;
writeln ('a=',a:7:2,'b=',b:7:2,'сумма введенных чисел
=',s:7:2);
end.
```

Окно вывода:

```
введите два числа
1 2
a=   1.00b=   2.00сумма введенных чисел =   3.00
```

Пример 2. Вычислить выражение

$$y = \frac{x}{1 - \frac{x^2}{3 - \frac{x^2}{5 - \frac{x^2}{7}}}}$$

Обозначения:
X – введенное число;
R, R1 – вспомогательные переменные;
Y – результат вычисления выражения.

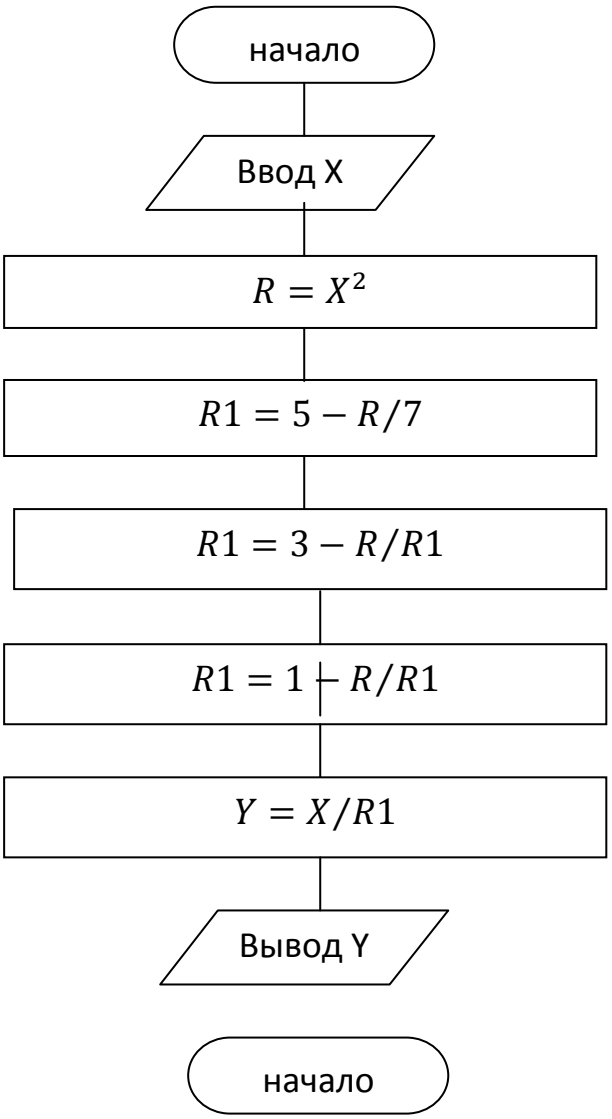


Рис. 2. Блок-схема алгоритма решения задачи Примера 2

Блок схема этого алгоритма (разобрана в пособии «Информатика. Основы алгоритмизации».. Пример 2.). Без комментариев она приведена на рис. 2.

Программа:

```
program primer_2;
var x,r,r1,y:real;
begin
writeln ('введите любое число');
readln(x);
r:=sqr(x);
r1:=5-r/7;
r1:=3-r/r1;
r1:=1-r/r1;
y:=x/r1;
```

```
writeln ('y=',y);
end.
```

Окно вывода:

```
введите любое число
1
y=1.55737704918033
```

Задания для самостоятельной работы

1. Даны любые пять чисел: a, b, c, d, e. Найти среднее арифметическое и среднее геометрическое заданных чисел.

2. Вычислить функцию
$z = x^5 - 0{,}2x^4 + 5{,}5x^3 + 0{,}1x - 2.5$ в точках $x = 1{,}7$; $x = -2{,}6$

3. Даны катеты прямоугольного треугольника a и b. Найти гипотенузу c и прилежащие углы прямоугольного треугольника α, β.

4. Известен диаметр окружности B. Требуется найти длину окружности L.

5. Известны два числа A и B, ни одно из них не равно нулю. Найти их сумму S, разность R, произведение P и частное. Результаты вывести на печать.

6. Известны катеты прямоугольного треугольника A и B. Требуется вычислить его гипотенузу C.

7. Вычислить функцию $y = 0{,}8x'^4 + 2{,}6x^5 - 2{,}3x^2 + x^6 + 1.57$ в точках $x = 1{,}1$; $x = 2{,}1$

8. Поменять местами содержимое ячеек A и B двумя способами.

9. Известны координаты точек A и B на плоскости: A(x1, y1), B(x2, y2).Требуется вычислить длину отрезка AB.

10. Известна диагональ квадрата A. Вычислить площадь S квадрата.

11. Известны три числовые величины A1, A2, A3. Требуется осуществить перенос содержимого величины A2 в A1, содержимого величины A3 в A2, содержимого величины A1 в A3. Результат перестановки вывести на печать.

12. Известна длина окружности L. Требуется вычислить радиус этой окружности R.

13. Известна сторона квадрата A. Вычислить длину диагонали D этого квадрата. Результат вывести на печать..

14. Даны три точки на координатной плоскости: A(x1, y1), B(x2, y2) и C(x3, y3). Требуется вычислить длины отрезков AB, BC и CA. Результат вычисления вывести на печать.

15. Известна гипотенуза c и один из прилежащих углов α прямоугольного треугольника. Найти площадь прямоугольного треугольника S и прилежащий угол β.

Условные операторы

Условные операторы предназначены для выбора к исполнению одного из возможных действий алгоритма в зависимости от результатов вычисления некого логического выражения, при этом одно из действий может отсутствовать. В паскале есть несколько условных операторов. В рамках этого пособия рассмотрим один из них – оператор if. Пусть на блок схеме ветвление имеет форму, изображенную на рис. 3.

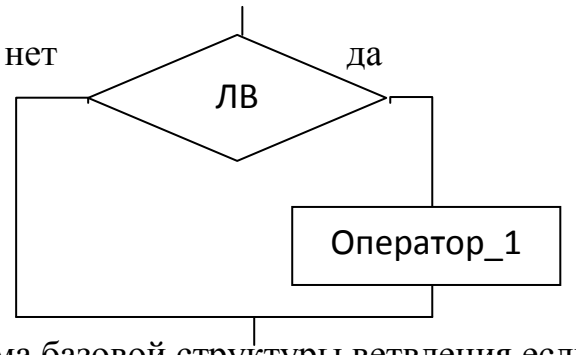


Рис. 3. Схема базовой структуры ветвления если-то

Здесь ЛВ – логическое выражение.

Описывается эта конструкция на Паскале следующим оператором:

If ЛВ then оператор_1;

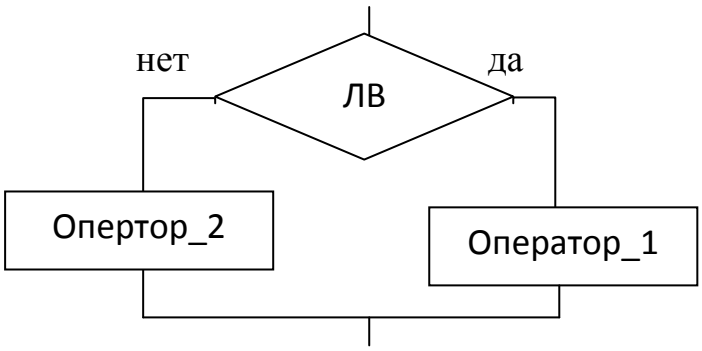


Рис. 4. Схема базовой структуры ветвления если-то-иначе

Если на блок схеме разветвляющаяся структура выглядит как на рис. 4, то описать ее средствами Паскаля можно с помощью оператора:

If ЛВ then оператор_1 else оператор_2;

Оператор if выполняется в таком порядке

1. Сначала вычисляется логическое выражение, записанное после слова if. В результате его вычисления получится величина логического типа (истина или ложь).

2. Если это истина, то выполняется оператор_1, записанный после слова then.

3. Если в результате вычисления логического выражения получим ложь. То выполняется опертор_2, записанный после слова else. Если в конструкции отсутствует слово else, то выполняется следующий оператор программы.

Если вместо оператора_1 или оператора_2 в конструкции ветвления стоит несколько операторов, то их нужно заключить в операторные скобки begin… end.

Конструкция, представляющая собой совокупность произвольного числа операторов, отделенных друг от друга точкой с запятой и ограниченную операторными скобками begin… end называют составным оператором. Он воспринимаете как единое целое в паскале, и может находиться в любом месте программы, где возможно наличие операторов. Например, в операторе if.

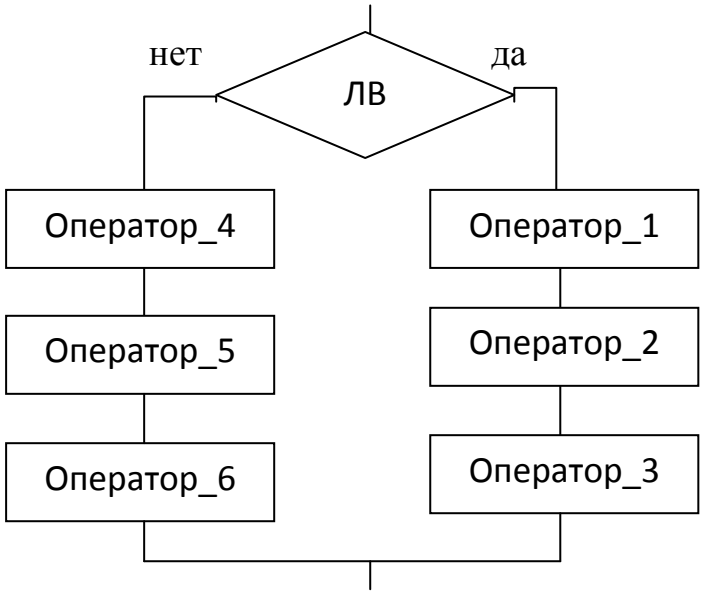


Рис. 5. Базовая структура ветвления, где при выборе одного из альтернативных путей решения приходится выполнять несколько действий

Например, конструкция, представленная на рис. 5 в паскале описывается оператором:

If ЛВ then begin оператор_1;оператор_2;оператор_3; end

else begin оператор_4;оператор_5;оператор_6; end;

В паскале точка с запятой может стоять только в конце оператора. И, если, в операторе, приведенном выше, опустить операторные скобки begin… end, то точка с запятой после оператора_1 спровоцирует конец оператора if и нарушит логику решения задачи. Чтобы избежать подобной ошибки и требуется выставлять операторные скобки. Скобка – не оператор, поэтому перед словом else точка с запятой не ставится. Она спровоцирует конец оператора if, кроме того будет синтаксической ошибкой т.к. нет в паскале оператора начинающегося со слова else.

Если структура ветвления переставлена в алгоритме в форме выбора (рис.6), то записать ее на паскале нужно так

if ЛВ-1 then оператор_1 else

if ЛВ-2 then оператор_2 else

if ЛВ-3 then оператор_3;

Здесь ЛВ 1 это логическое выражение 1, ЛВ 2 это логическое выражение 2, ЛВ 3 это логическое выражение 3.

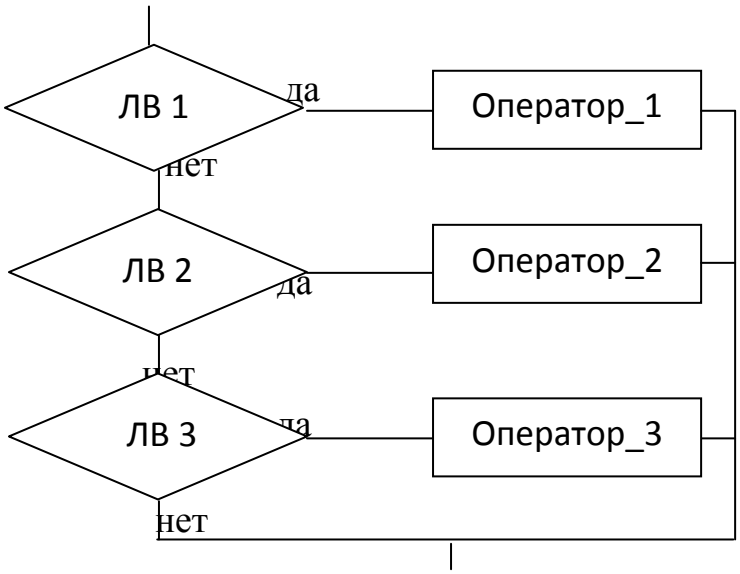


Рис. 6. Базовая структура ветвления в виде выбора

Следует обратить внимание, что точка с запятой стоит только в конце оператора, т.е. только там, где все ветви, образовавшиеся в процессе ветвления – соединились (пришли в одну точку, т.е. конструкция ветвления закончилась).

Читается конструкция так:

Если ЛВ-1 является истиной, то выполнить опертор-1, иначе

если ЛВ-2 будет истиной, то выполнить оператор 2, иначе

если ЛВ-3 в результате вычисления будет истиной, то выполнить оператор_3 в противном случае ничего не делать. Конец оператора (конструкции) ветвления.

Таким образом, какова бы не была глубина вложенности оператора if. точка с запятой ставится только в конце структуры ветвления.

Если структура ветвления имеет форму, представленную на рис. 7, то на паскале этот оператор запишем так

if ЛВ-1 then begin оператор_1;оператор_2; end else

if ЛВ-2 then begin оператор_3;оператор_4; end else

if ЛВ-3 then begin оператор_5;оператор_6; end else

begin оператор_7;оператор_8; end;

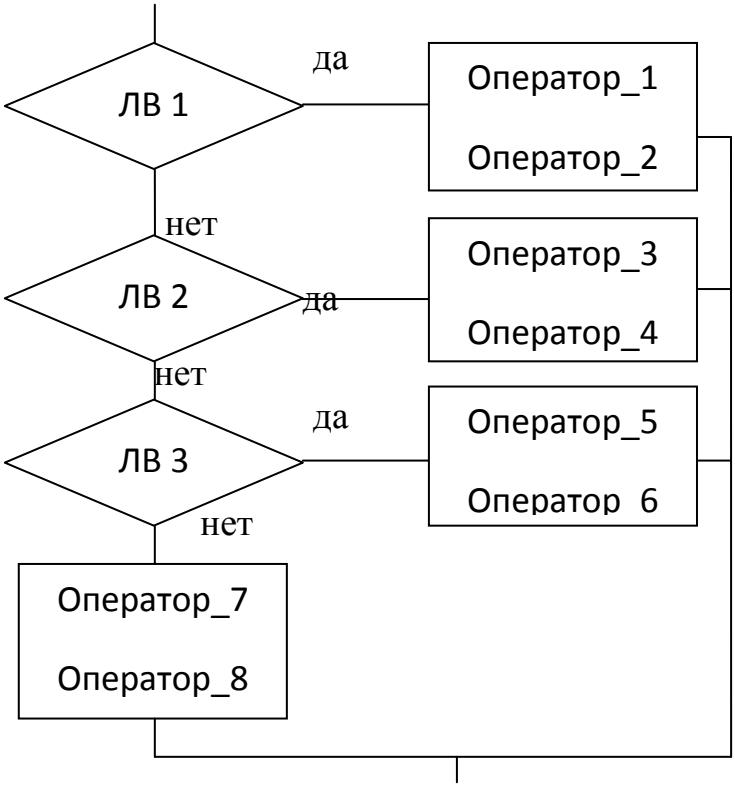


Рис. 7. Базовая структура ветвления в виде выбора, где в случае выбора одного из альтернативных путей решения требуется выполнить несколько операторов

Пример 3. Вводится число, если оно положительное требуется его удвоить.

Математическая модель:

$$x = 2x \qquad если \qquad x > 0$$

x – любое введенное число

Блок-схема алгоритма (разобрана в пособии "«Информатика. Основы алгоритмизации» Пример 3. Без пояснений она приведена на рис. 8.

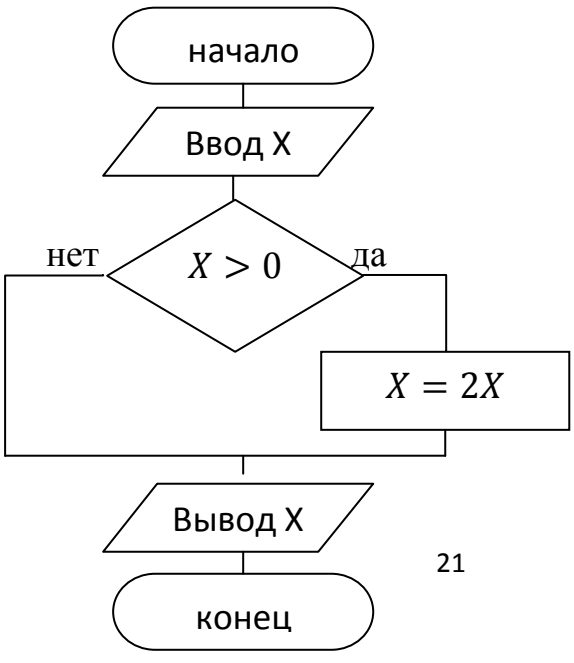


Рис. 8. Блок-схема алгоритма решения задачи в примере 3

Программа:

```
program primer_3;
var x:real;
begin
writeln ('введите любое число');
readln (x);
if x>0 then x:=2*x;
writeln ('x=',x);
end.
```

Протестируем программу на различных числах, больших, меньших и равных нулю.

Окно вывода (случай, если введенное число положительное):

```
введите любое число
5
x=10
```

Окно вывода (случай, если введенное число отрицательное):

```
введите любое число
-7
x=-7
```

Окно вывода (случай, если введенное число ноль):

```
введите любое число
0
x=0
```

Пример 4. Вводится число, если оно положительное требуется его удвоить, в противном случае утроить. Разработать блок схему алгоритма.

Математическая модель:

$$X = \begin{cases} 2X, & если \quad X > 0 \\ 3X, & если \quad X \leq 0 \end{cases}$$

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 4. Без пояснений она приведена на рис. 9.

Программа:

```
program primer_4;
var x:real;
begin
writeln ('введите любое число');
readln (x);
if x>0 then x:=2*x else x:=3*x;
writeln ('x=',x);
end.
```

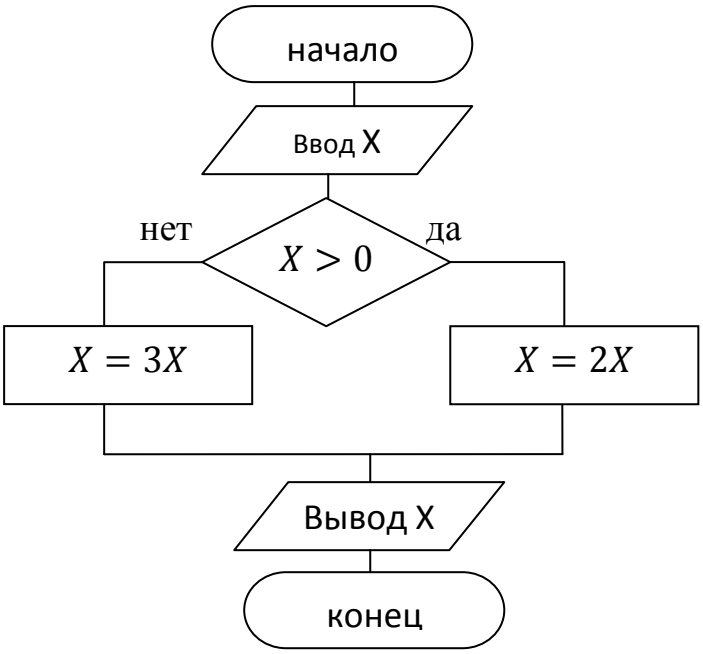


Рис. 9. Блок-схема алгоритма решения задачи в примере 4

Окно вывода (случай, если введенное число положительное):

```
введите любое число
4
x=8
```

Окно вывода (случай, если введенное число отрицательное):

```
введите любое число
-1
x=-3
```

Окно вывода (случай, если введенное число ноль):

```
введите любое число
0
x=0
```

Пример 5. Вводится число, если оно положительное требуется его удвоить, если отрицательное утроить, если это ноль оставить без изменения.

Математическая модель:

$$X = \begin{cases} 2X, & если \quad X > 0 \\ 3X, & если \quad X < 0 \\ X, & если \quad X = 0 \end{cases}$$

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 5. Без пояснений она приведена на рис. 10.

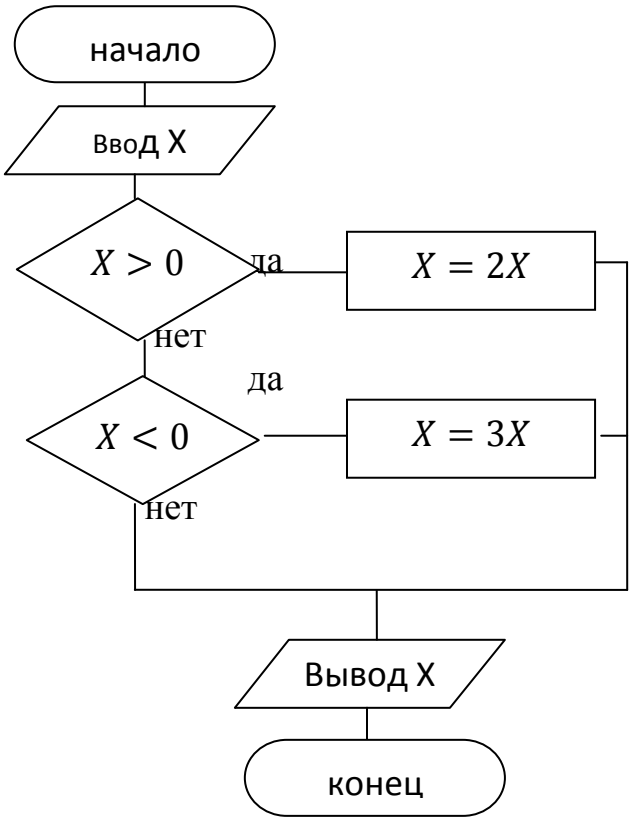


Рис. 10. Блок-схема алгоритма решения задачи примера 5

Программа:

```
program primer_5;
var x:real;
begin
writeln ('введите любое число');
readln (x);
if x>0 then x:=2*x else
if x<0 then x:=3*x;
writeln ('x=',x);
end.
```

Окно вывода (случай, если введенное число положительное):

```
введите любое число
10.5
x=21
```

Окно вывода (случай, если введенное число отрицательное):

```
введите любое число
-3.3
x=-9.9
```

Окно вывода (случай, если введенное число ноль):

```
введите любое число
0
x=0
```

Пример 6. Вводится число, если оно положительное требуется его удвоить, если отрицательное утроить, если это ноль увеличить на единицу.

Математическая модель:

$$X = \begin{cases} 2X, & \textit{если} \quad X > 0 \\ 3X, & \textit{если} \quad X < 0 \\ X + 1, & \textit{если} \quad X = 0 \end{cases}$$

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 6. Без пояснений она приведена на рис. 11.

Программа:

```
program primer_6;
var x:real;
begin
writeln ('введите любое число');
readln (x);
if x>0 then x:=2*x else
if x<0 then x:=3*x else
x:=x+1;
writeln ('x=',x);
end.
```

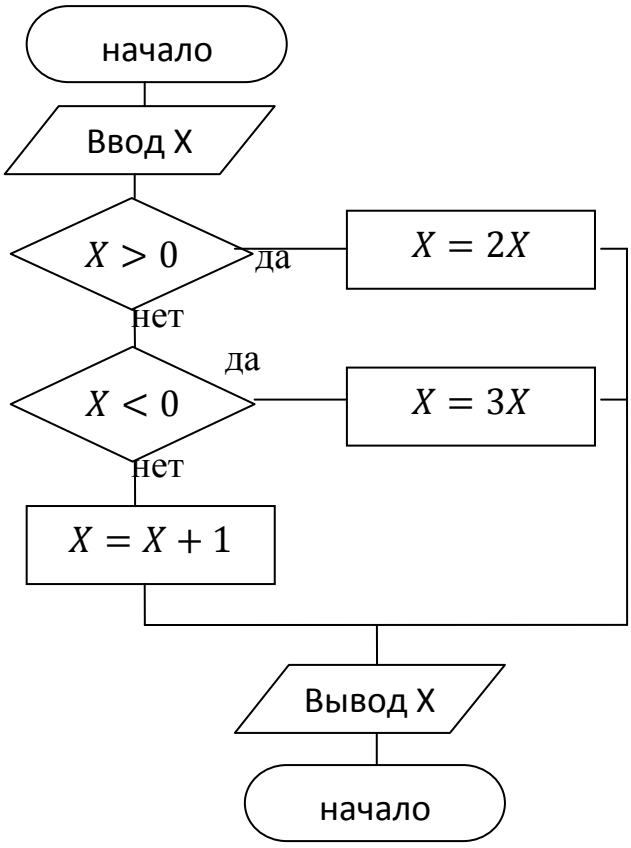


Рис. 11. Блок-схема алгоритма решения задачи примера 6

Окно вывода (случай, если введенное число положительное):

```
введите любое число
2.2
x=4.4
```

Окно вывода (случай, если введенное число отрицательное):

```
введите любое число
-6.5
x=-19.5
```

Окно вывода (случай, если введенное число ноль)

```
введите любое число
0
x=1
```

Пример 7. Вводится число, если оно положительное, требуется его удвоить, если отрицательное утроить.

Математическая модель:

$$X = \begin{cases} 2X, & \textit{если} \quad X > 0 \\ 3X, & \textit{если} \quad X < 0 \end{cases}$$

Блок-схема алгоритма (разобрана в пособии «Информатика. Основы алгоритмизации». Пример 7. Блок-схема без пояснений приведена на рис. 12.

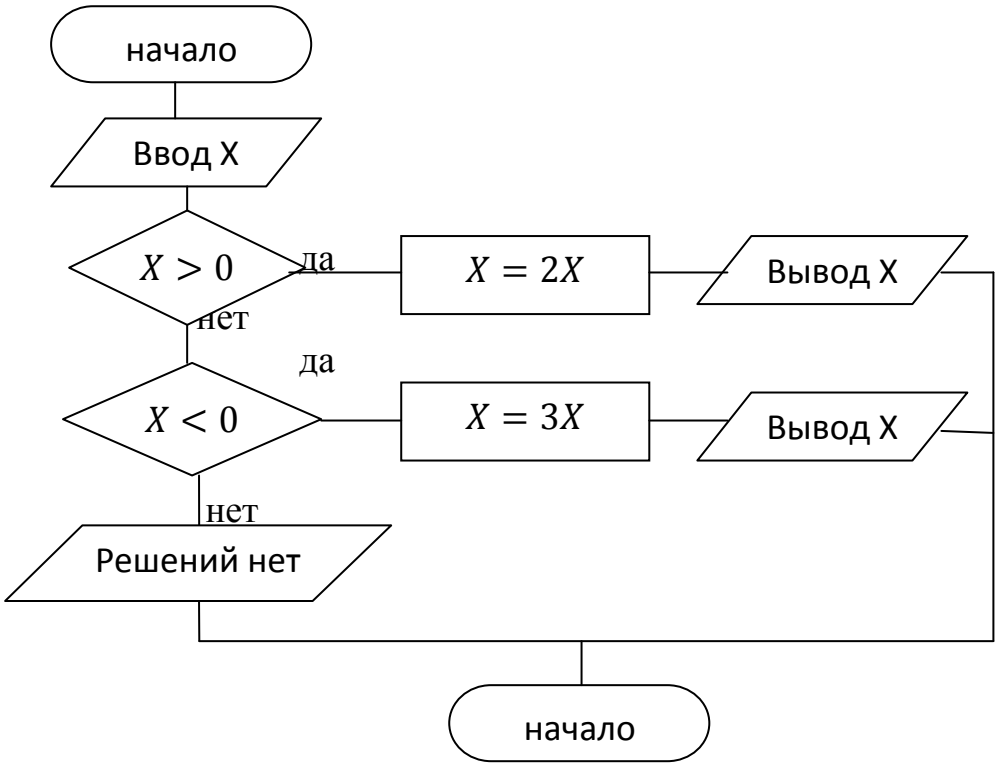


Рис 12. Блок-схема алгоритма решения задачи примера 7

Программа:

```
program primer_7;
var x:real;
begin
writeln ('введите любое число');
```

```
readln (x);
if x>0 then begin x:=2*x; writeln ('x=',x); end
else
if x<0 then begin x:=3*x; writeln ('x=',x); end
else
writeln('');
end.
```

Окно вывода (случай, если введенное число положительное):

```
введите любое число
3.1
x=6.2
```

Окно вывода (случай, если введенное число отрицательное):

```
введите любое число
-6
x=-18
```

Окно вывода (случай, если введенное число ноль):

```
введите любое число
0
решений нет
```

Пример 8. Найти максимальное из трех заданных чисел.

Обозначения:

X,Y,Z – введенные числа;

MAX – максимальное из чисел.

Алгоритм 1. Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации» Пример 8. Блок-схема без пояснений приведена на рис. 13.

Программа:

```
program primer_8_1;
var x,y,z,max:real;
begin
writeln ('введите любые три числа');
readln (x,y,z);
if x>y then  max:=x else max:=y;
if z>max then max:=z;
writeln('max=',max);
end.
```

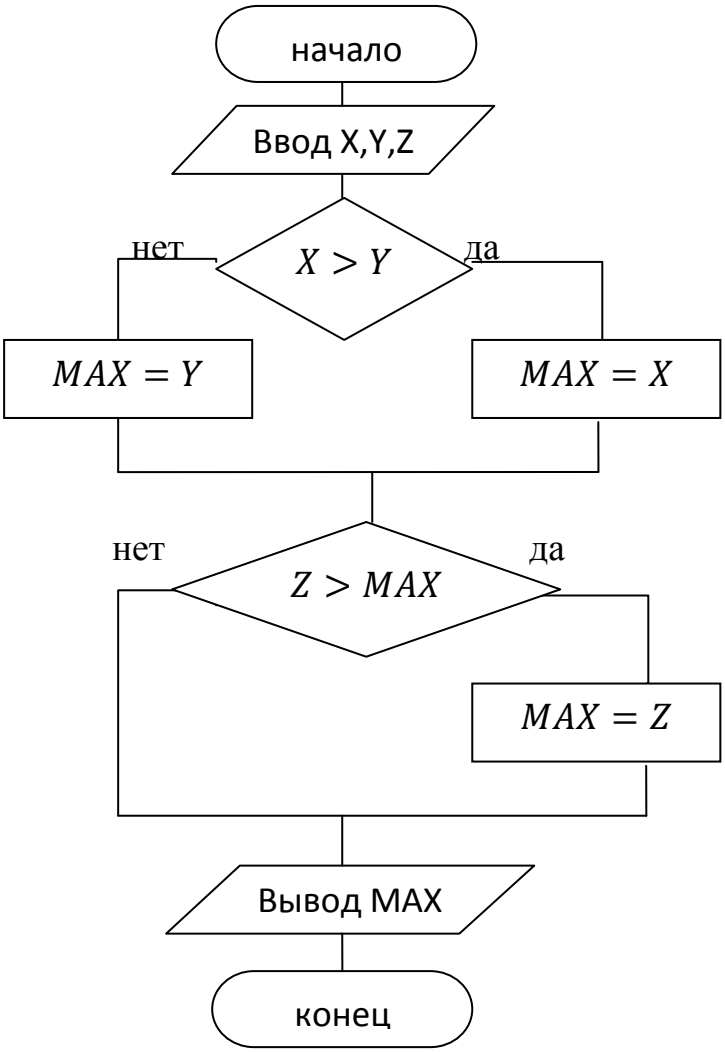


Рис. 13. Блок-схема алгоритма решения задачи примера 8

Окно вывода (случай, если числа вводятся по возрастанию):

```
введите любые три числа
1 2 3
max=3
```

Окно вывода (случай, если числа вводятся по убыванию):

```
введите любые три числа
3 2 1
max=3
```

Окно вывода (случай, если числа вводятся в произвольном порядке):

```
введите любые три числа
1 3 2
max=3
```

Окно вывода (случай, если числа вводятся одинаковые):

```
введите любые три числа
1 1 1
max=1
```

Алгоритм 2. Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 8. Блок-схема без пояснений приведена на рис. 14.

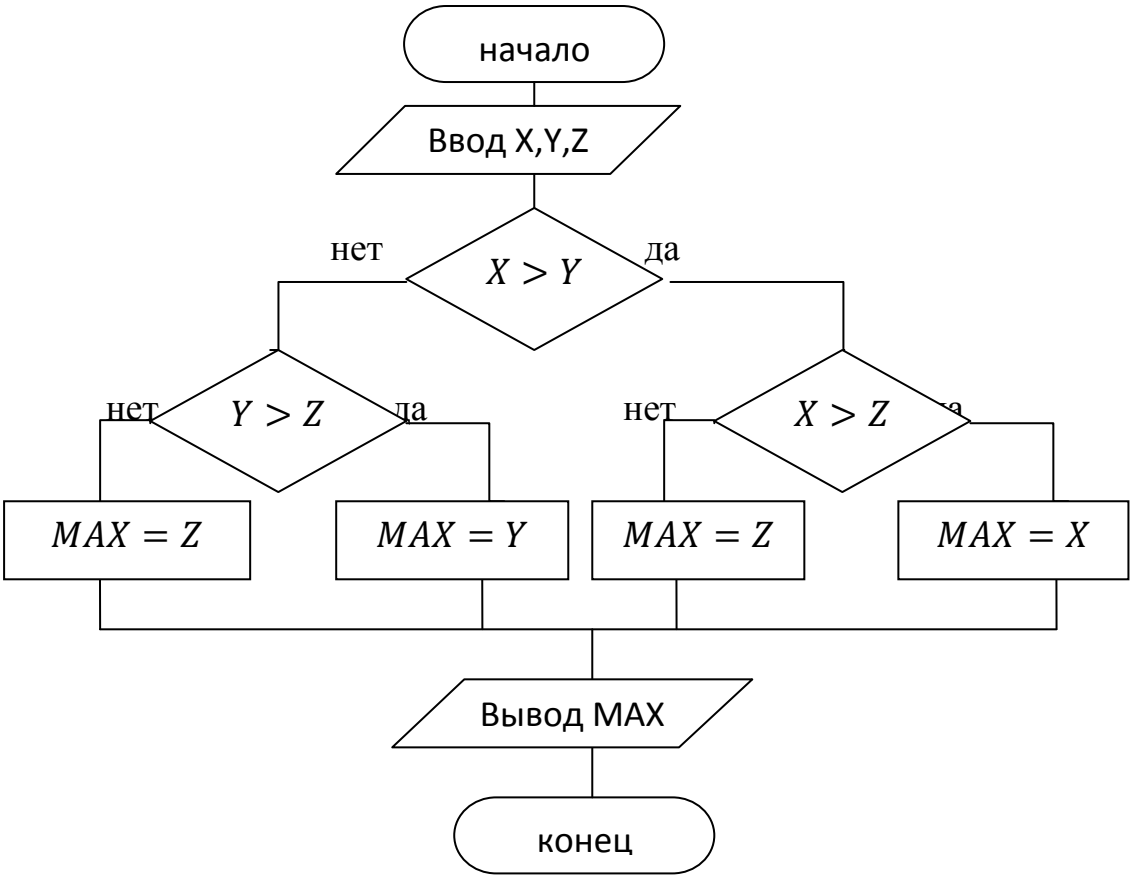


Рис. 14. Блок-схема алгоритма решения задачи примера 8

Программа:

```
program primer_8_2;
var x,y,z,max:real;
begin
writeln ('введите любые три числа');
readln (x,y,z);
if x>y then  if x>z then max:=x else max:=z
       else  if y>z then max:=y else max:=z;
writeln('max=',max);
end.
```

Окно вывода (случай, если числа вводятся по возрастанию):

```
введите любые три числа
1 2 3
max=3
```

Окно вывода (случай, если числа вводятся по убыванию):

```
введите любые три числа
3 2 1
max=3
```

Окно вывода (случай, если числа вводятся в произвольном порядке):

```
введите любые три числа
1 3 2
max=3
```

Окно вывода (случай, если числа вводятся одинаковые):

```
введите любые три числа
1 1 1
max=1
```

Пример 9. Упорядочить три числа по возрастанию.

Обозначения:

X,Y,Z – введенные числа;

K – вспомогательная переменная (для хранения копии числа при обмене содержимого двух ячеек).

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 9. Блок-схема без пояснений на рис. 15.

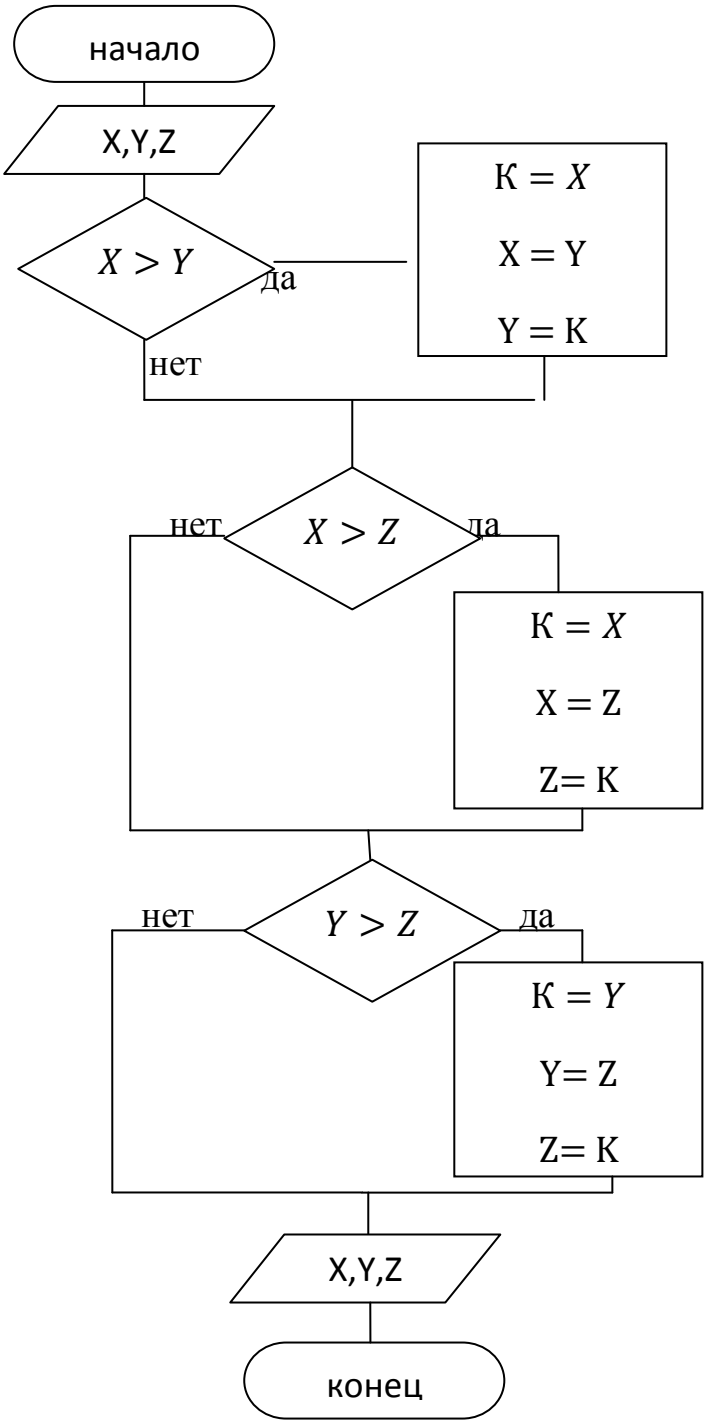


Рис. 15. Блок-схема алгоритма решения задачи примера 9

Программа:

```
program primer_9;
```

```
var x,y,z,k:real;
begin
writeln ('введите любые три числа');
readln (x,y,z);
if x>y then  begin k:=x;x:=y;y:=k;end;
if x>z then  begin k:=x;x:=z;z:=k;end;
if y>z then  begin k:=y;y:=z;z:=k;end;
writeln('x=',x,'y=',y,'z=',z);
end.
```

Окно вывода (случай, если числа вводятся по возрастанию):

```
введите любые три числа
1.1 2.2 3.3
x=1.1  y=2.2   z=3.3
```

Окно вывода (случай, если числа вводятся по убыванию):

```
введите любые три числа
3.3 2.2 1.1
x=1.1  y=2.2   z=3.3
```

Окно вывода (случай, если числа вводятся в произвольном порядке):

```
введите любые три числа
1.1 3.3 2.2
    x=1.1  y=2.2   z=3.3
```

Окно вывода (случай, если числа вводятся одинаковые):

```
введите любые три числа
1.1 1.1 1.1
x=1.1  y=1.1   z=1.1
```

Пример 10. Вычислить

$$y = \begin{cases} x, & \textit{если} \quad -3 < x < 3 \\ x^2, & \textit{если} \quad 3 \le x < 5 \\ x^3, & \textit{если} \quad x \ge 5 \end{cases}$$

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 10. Блок-схема без пояснений приведена на рис. 16.

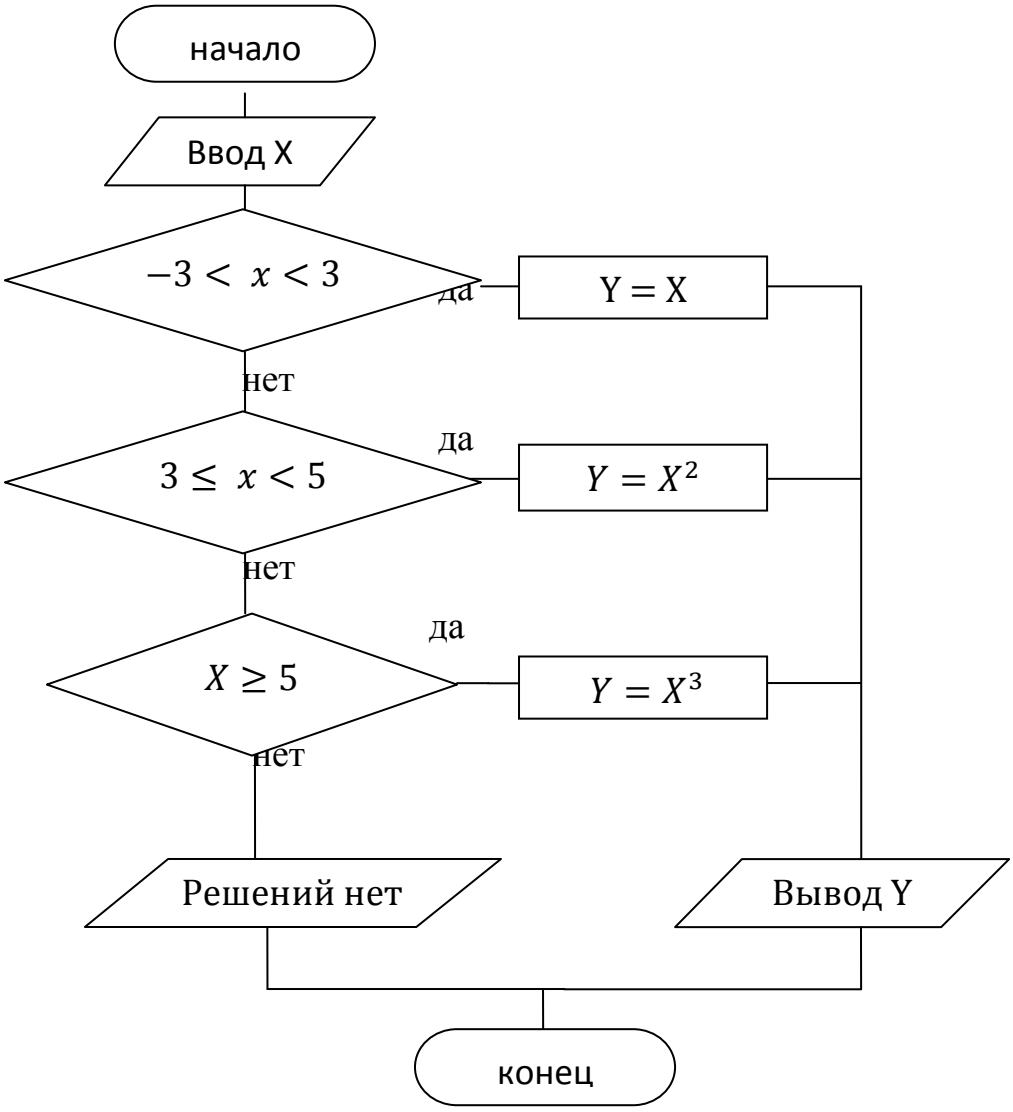


Рис. 16. Блок-схема алгоритма решения задачи примера 10

Программа:

```
program primer_10_1;
var x,y:real;
begin
writeln ('введите любое число');
```

```
readln (x);
if (x>-3)and(x<3) then
    begin
        y:=x;
        writeln('y=',y);
    end
else
if (x>=3)and(x<5) then
    begin
        y:=sqr(x);
        writeln('y=',y);
    end
else
if x>=5 then
    begin
        y:=exp(3*ln(x));
        writeln('y=',y);
    end
else
writeln('решений нет');
end.
```

Окно вывода (случай, если введенное число принадлежит промежутку от минус бесконечности до -3 включительно):

```
введите любое число
-6.1
решений нет
```

Окно вывода (случай, если введенное число принадлежит промежутку от минус трех до трех, не включая концов промежутка):

```
введите любое число
0
y=0
```

Окно вывода (случай, если введенное число принадлежит промежутку от трех включительно до 5, не включая правый конец промежутка):

```
введите любое число
3
y=9
```

Окно вывода (случай, если введенное число окажется более или равно 5):

```
введите любое число
5
y=125
```

Или алгоритм (блок-схема без пояснения на рис. 17) отличается от предыдущего, только порядком проверки условий

Программа:

```
program primer_10_2;
var x,y:real;
begin
writeln ('введите любое число');
readln (x);
if x>=5 then
    begin
        y:=exp(3*ln(x));
        writeln('y=',y);
    end
else
if (x>=3) then
    begin
        y:=sqr(x);
        writeln('y=',y);
    end
else
if x>-3 then
    begin
        y:=x;
        writeln('y=',y);
    end
else
writeln('решений нет');
end.
```

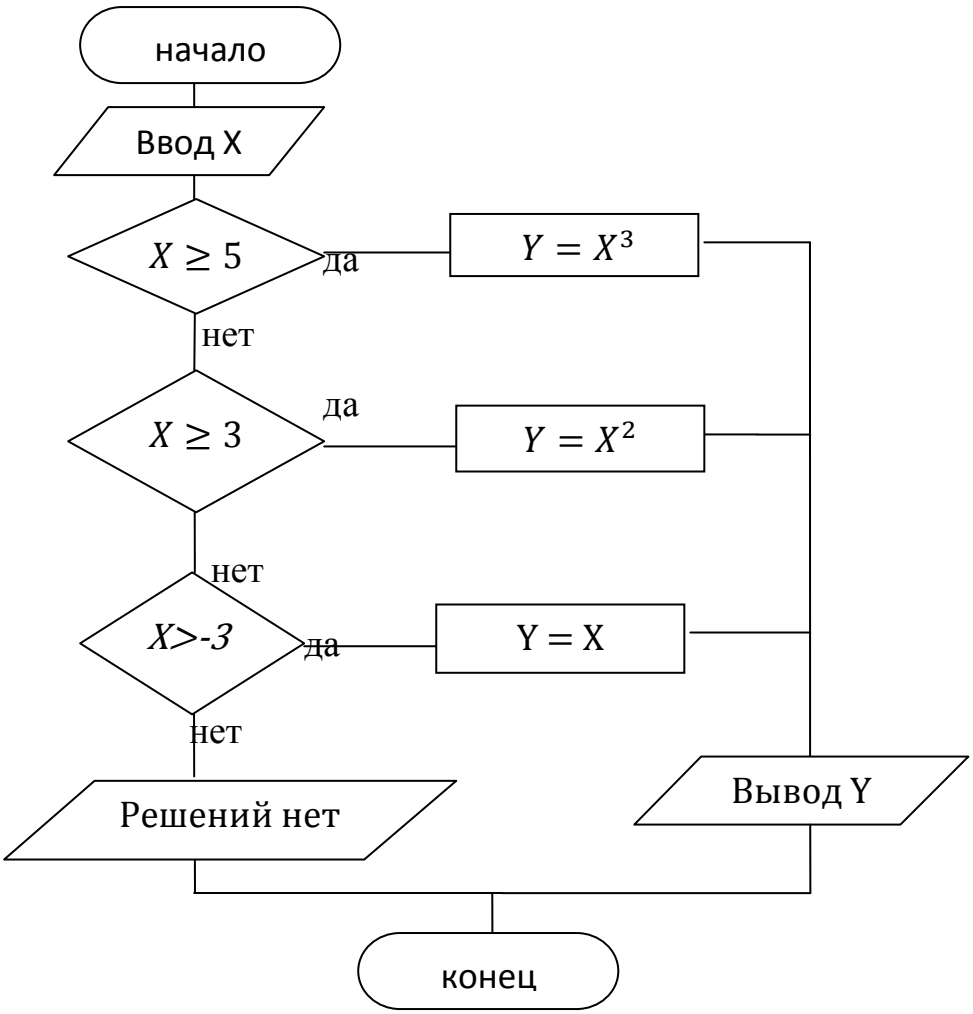


Рис. 17. Блок-схема алгоритма решения задачи примера 10

При тестировании, конечно, получим те же результаты.

Пример 11. Даны вещественные числа x и y. Определить, принадлежит ли точка с координатами (x; y) заштрихованной области (рис. 18.).

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 11. Блок-схема без пояснений приведена на рис. 19.

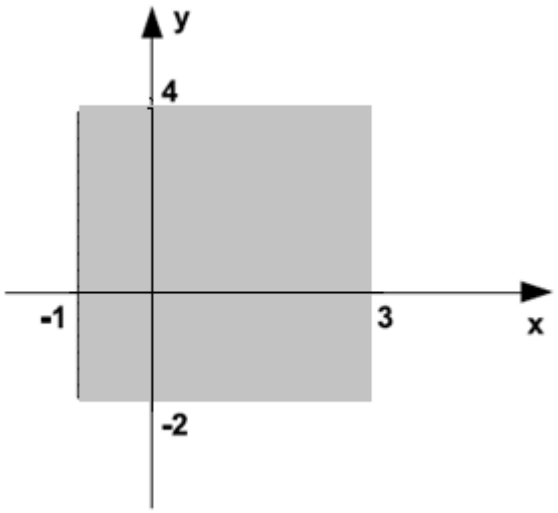


Рис. 18.

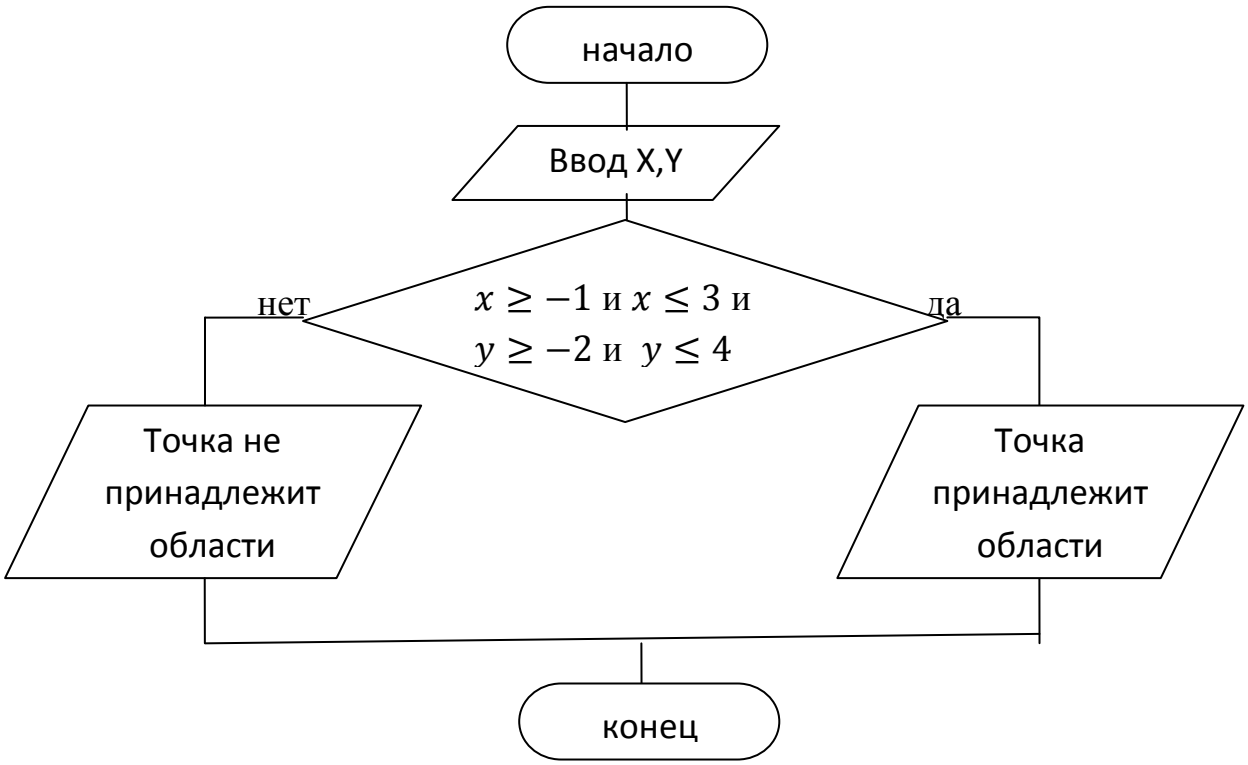


Рис. 19. Блок-схема алгоритма решения задачи примера 11

Программа:

```
program primer_7;
var x,y:real;
begin
writeln ('введите координаты точки на
плоскости');
readln (x,y);
```

```
if (x>=-1)and(x<=3)and(y>=-2)and(y<=4) then
writeln('точка принадлежит заданной области')
else
writeln('точка не принадлежит заданной
области');
end.
```

Окно вывода (точка в области):

```
введите координаты точки на плоскости
0 0
точка принадлежит заданной области
```

Окно вывода (точка вне области):

```
введите координаты точки на плоскости
-5 -5
точка не принадлежит заданной области
```

Окно вывода (точка вне области):

```
введите координаты точки на плоскости
1 6
точка не принадлежит заданной области
```

Окно вывода (точка на границе области):

```
введите координаты точки на плоскости
2 -2
точка принадлежит заданной области
```

Задания для самостоятельной работы

1. Дано вещественное число B. Вычислить значения функции в заданной точке $y = f(B)$. Графики функции $y = f(B)$ приведены на рис. 20-43.

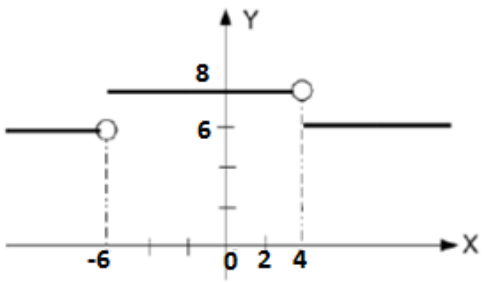


Рис. 20

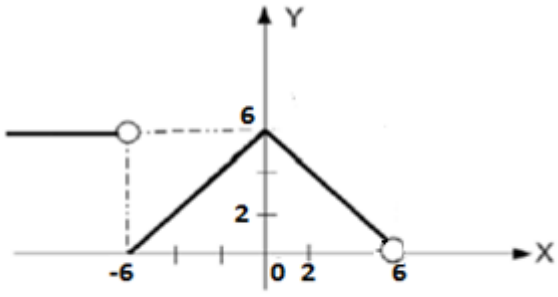


Рис. 21

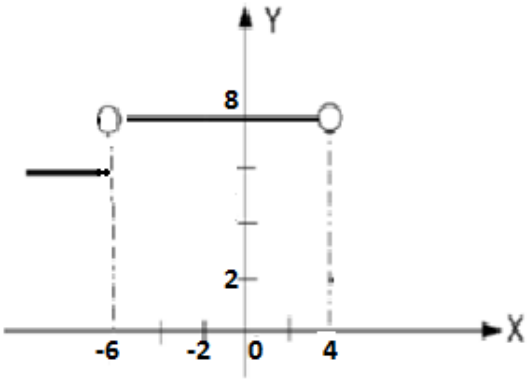


Рис. 23

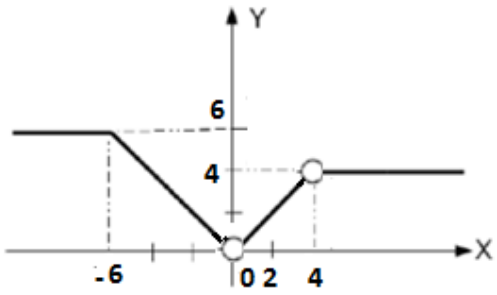


Рис. 22

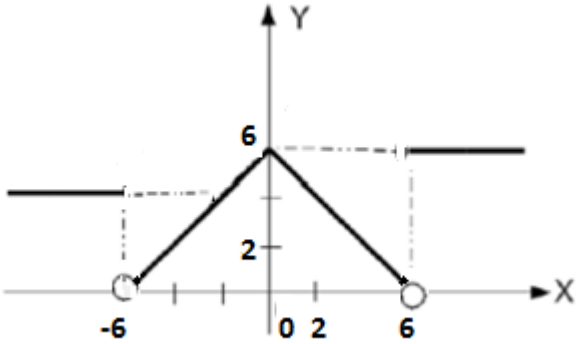


Рис. 24

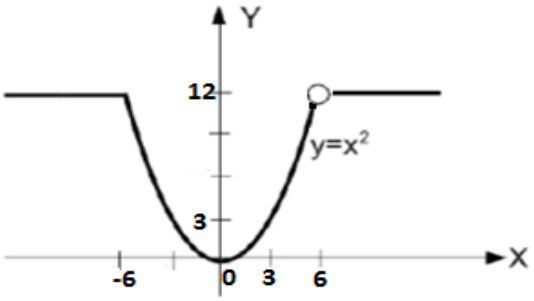


Рис. 25

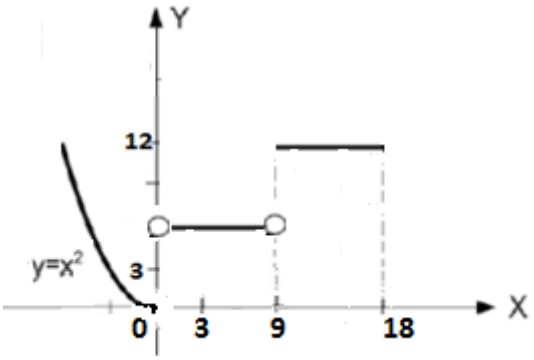


Рис. 26

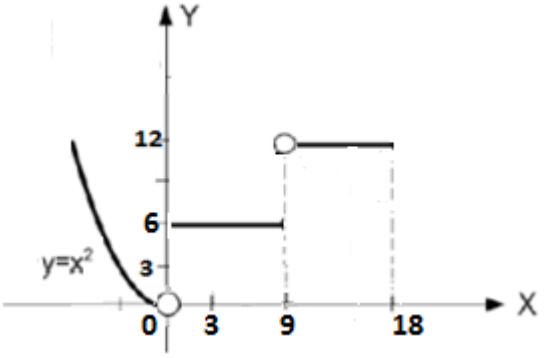


Рис. 27

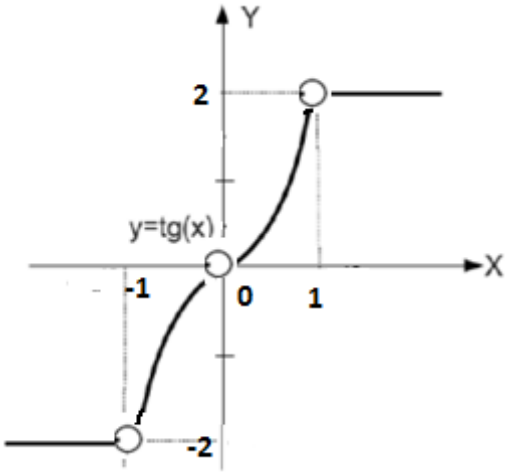


Рис. 29

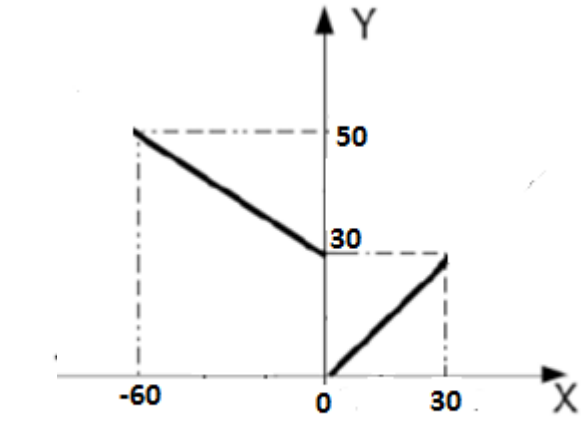


Рис. 28

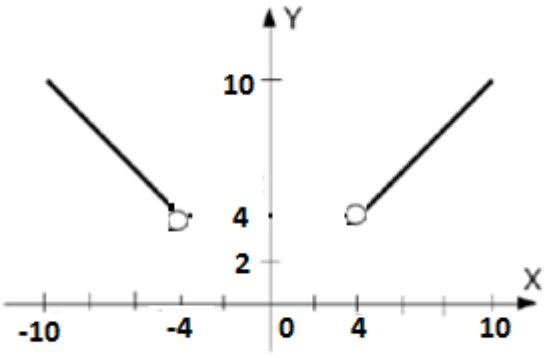


Рис. 30

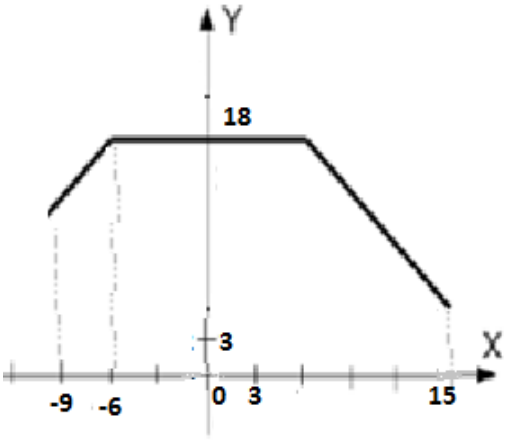
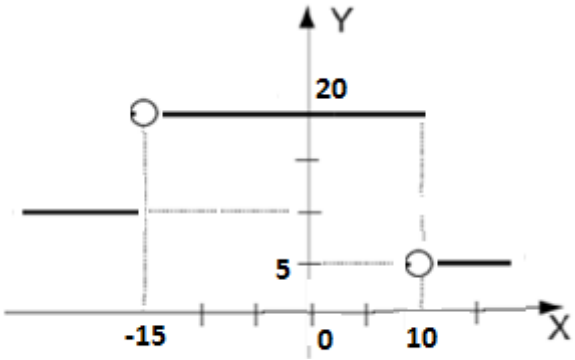


Рис. 31



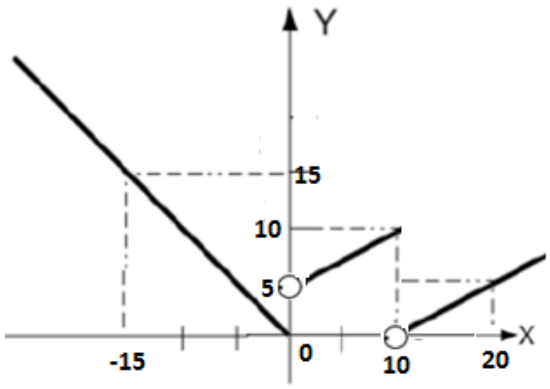


Рис. 32

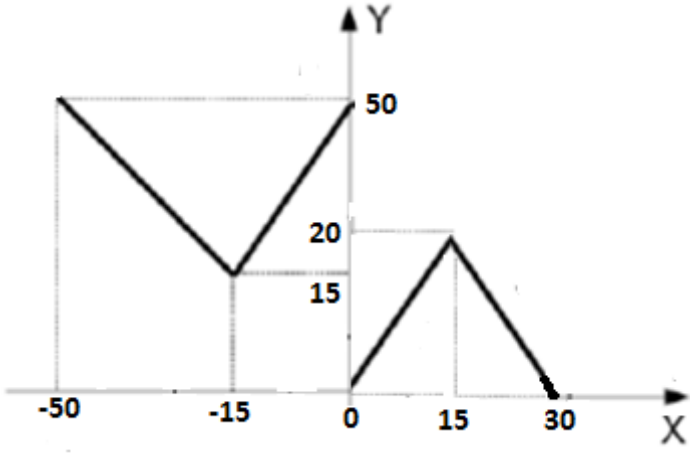


Рис. 33

Рис. 34

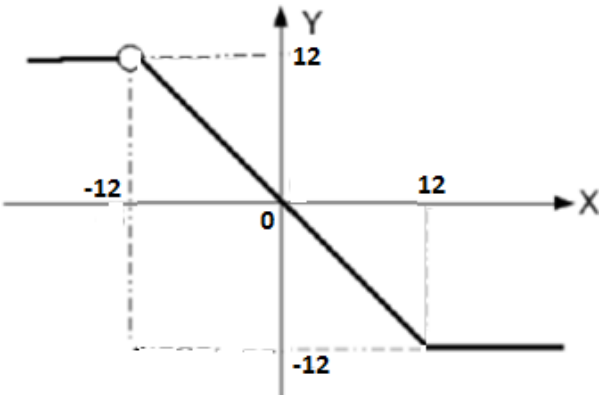


Рис. 35

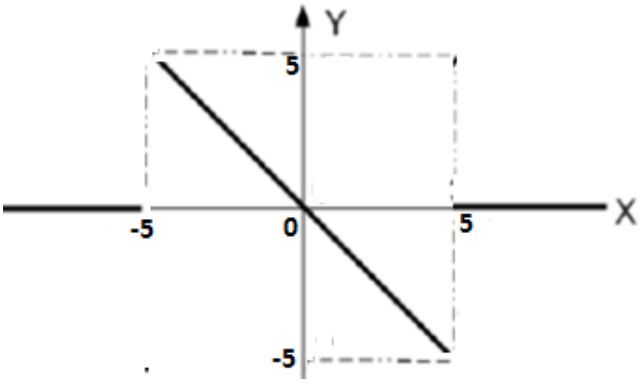


Рис. 36

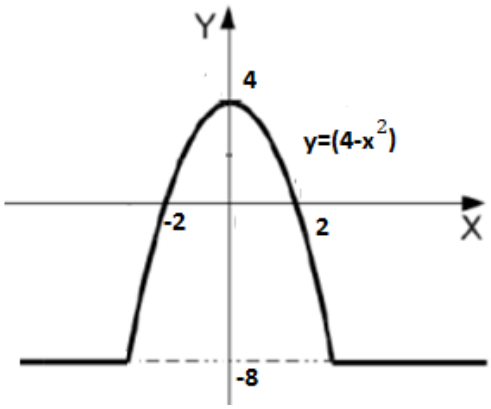


Рис. 37

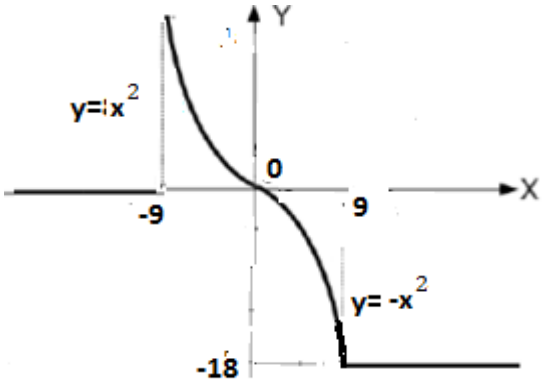
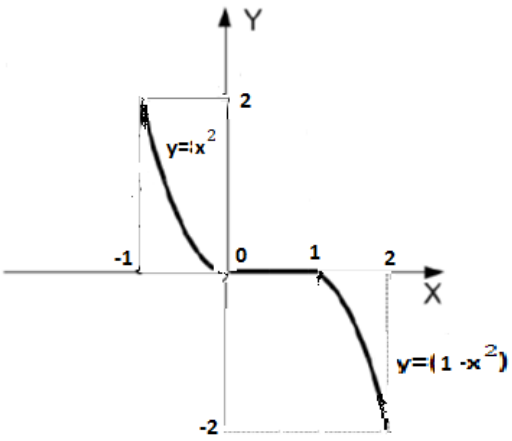


Рис. 38

Рис. 39



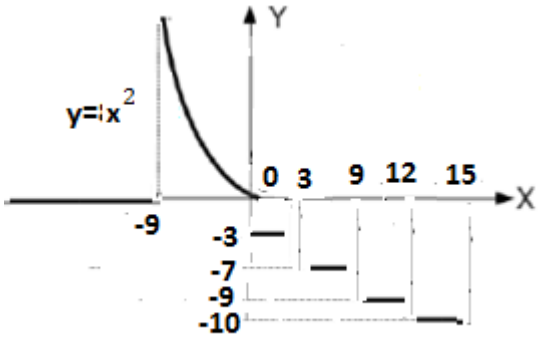


Рис. 40

Рис. 41

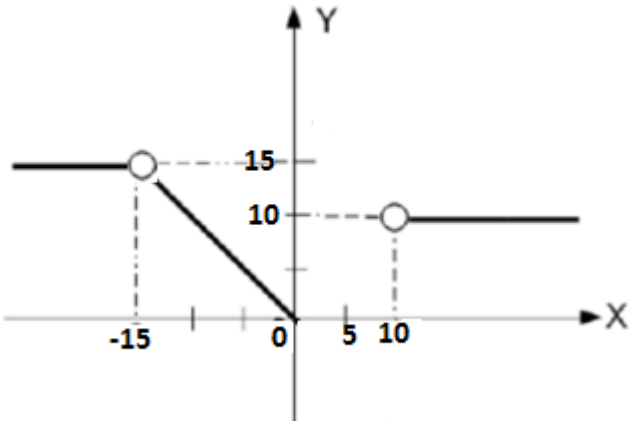


Рис. 42

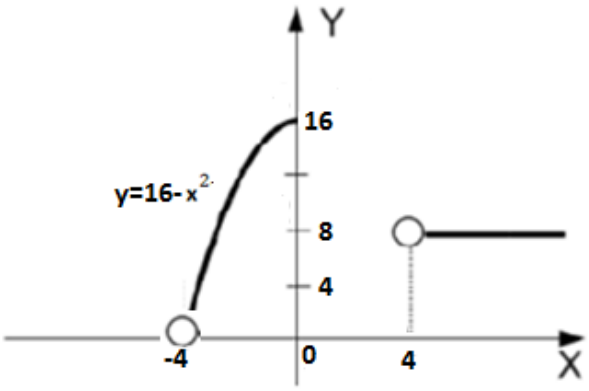


Рис. 43

2. Дана точка А с координатами (x,y) на плоскости. Определить, принадлежит ли эта точка закрашенной области. Варианты заданий на рис. 44-52.

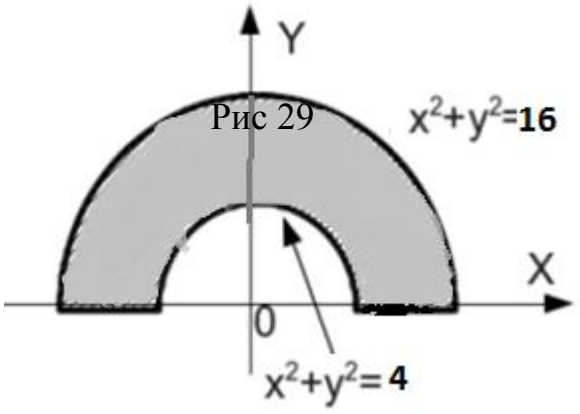


Рис 29

Рис. 44

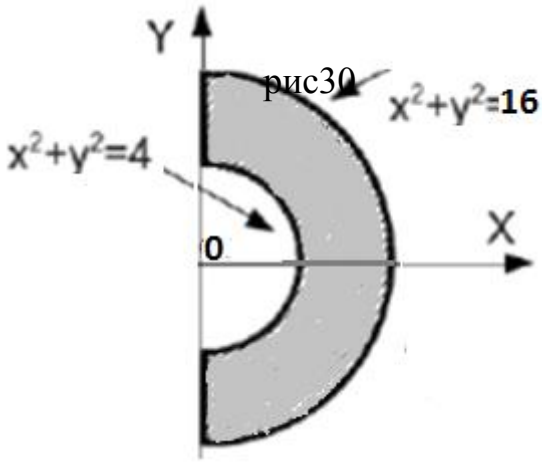
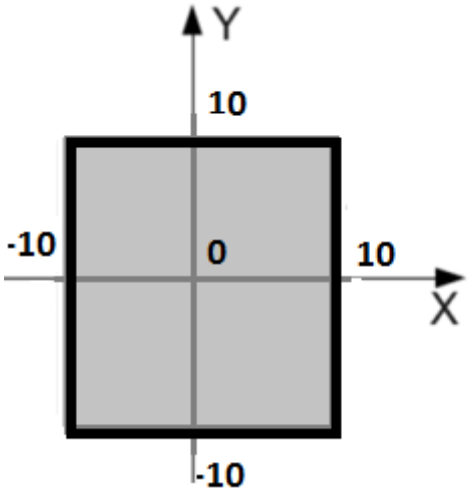


рис30

Рис. 45



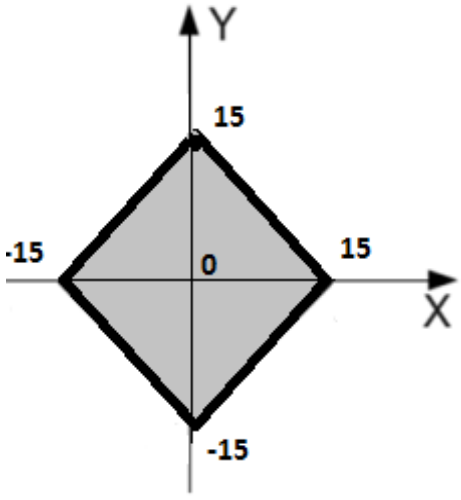


Рис. 46

Рис. 47

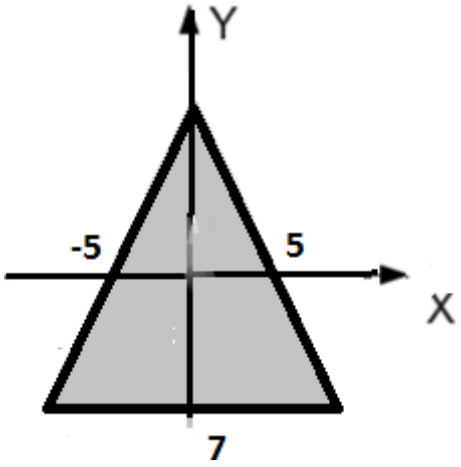


Рис. 48

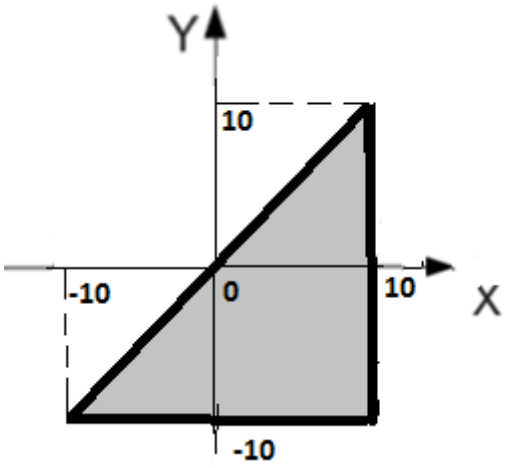


Рис. 49

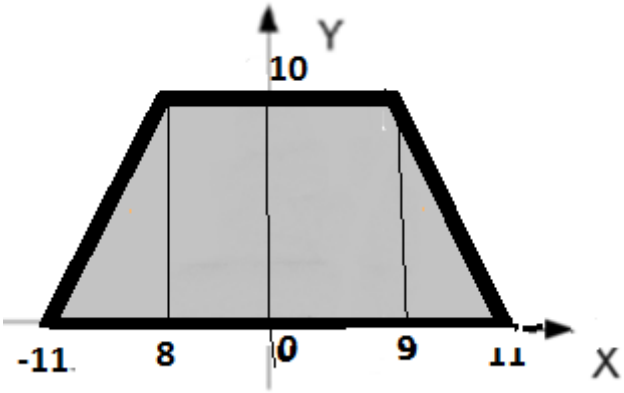


Рис. 50

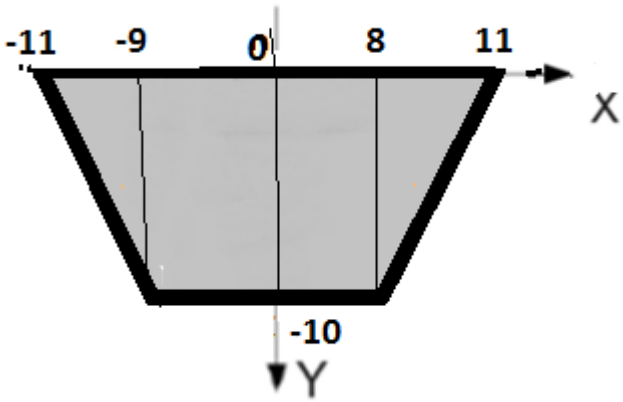


Рис. 51

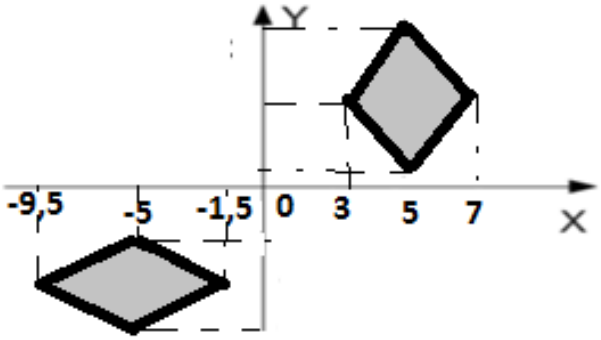


Рис. 52

3. Задана окружность. Центр окружности – в точке (x, y), ее радиус – r. Определить, пересекается ли заданная окружность с осью абсцисс и если пересекается – найти координаты точек пересечения.

4. Задана окружность. Центр окружности – в точке (x, y), ее радиус – r. Определить, пересекается ли заданная окружность с осью ординат, и если пересекается – найти координаты точек пересечения.

5. Вычислить функцию:

$$Z = \begin{cases} \dfrac{3X+5Y}{X^2+Y^2}, & если \quad x^2+y^2 \geq 1 \\ 7X^3+2Y+X, & если \quad x^2+y^2<1, \quad x \geq 0 \\ 3Y^3-5Y+X, & если \quad x^2+y^2<1, \quad x<0 \end{cases}$$

6. Вычислить функцию:

$$Y = \begin{cases} 3lg\left(\dfrac{x}{2}+1\right), & если \quad x>2 \\ e^x-2x^2, & если \quad x<2 \end{cases}$$

7. Задана окружность с центром в точке O(x0, y0) и радиусом R0. Определить, пересекается ли заданная окружность с осью абсцисс, если пересекается - найти точки пересечения.

8. Задана окружность с центром в точке O(x0, y0) и радиусом R0. Определить, пересекается ли заданная окружность с осью ординат, если пересекается – найти точки пересечения.

9. Вычислить:

$$Z = \begin{cases} \dfrac{3X+5Y}{X^2+Y^2}, & если \quad x^2+y^2 \geq 1 \\ 7X^3+2Y+X, & если \quad x^2+y^2<1, \quad x \geq 0 \\ 3Y^3-5Y+X, & если \quad x^2+y^2<1, \quad x<0 \end{cases}$$

10. Вычислить:

$$Y = \begin{cases} 3lg\left(\dfrac{x}{2}+1\right), & если \quad x>2 \\ e^x-2x^2, & если \quad x<2 \end{cases}$$

Циклический алгоритм

В Паскале различают три типа операторов цикла. Цикл с предшествующим условием, реализуется при помощи оператора цикла **while**. Цикл с последующим условием реализуется оператором цикла **repeat**. Цикл с параметром – оператором цикла **for**.

Оператор while

Если в программе необходимо описать базовую алгоритмическую структуру, представленную на рис. 53, то в паскале она описывается оператором цикла while.

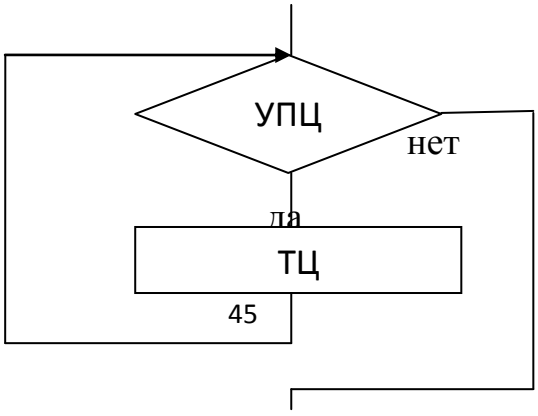


Рис. 53. Схема базовой алгоритмической структуры цикл с предшествующим условием

Общий вид оператора:

while УПЦ do ТЦ;

Здесь:
УПЦ – условие продолжения цикла;

ТЦ – тело цикла.

Читается эта запись так - пока УПЦ – истина - делать ТЦ.

Если операторов в теле цикла более одного, то эту серию операторов нужно заключить в операторные скобки begin…end для того, чтобы точка с запятой после первого оператора в теле цикла не спровоцировала завершение цикла.

Порядок выполнения оператора while:

1. Вычисляется УПЦ.

2. Если при вычислении выражения в пункте 1 получится истина, то выполняется ТЦ

3. Все циклически повторяется, начиная с пункта 1.

4. Если в результате вычисления выражения в пункте 1 получится ложь – цикл завершается. Управление программой передается следующему после оператора цикла оператору.

Чтобы избежать зацикливания необходимо следить, чтобы как минимум одна из величин присутствующих в условии продолжения цикла меняла свое значение в теле цикла.

Пример 13. Вычислить функцию $y = x^2$ в точках интервала $x \in [a.b]$, точки на интервале выбираются с заданным шагом h.

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 13. Блок-схема без пояснений приведена на рис. 54.

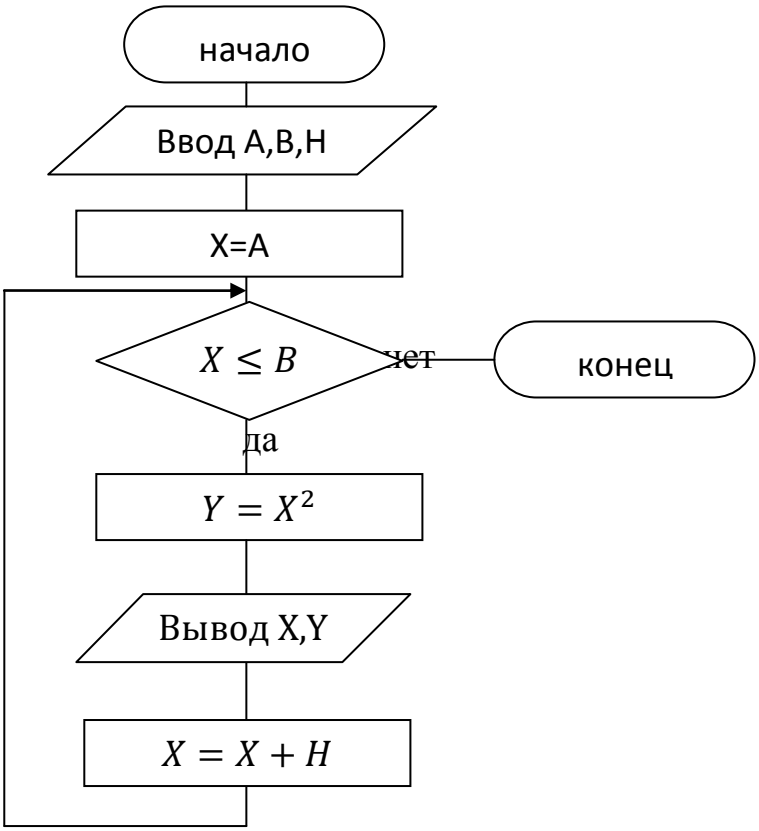


Рис. 54. Блок-схема алгоритма решения задачи примера 13

Обозначения:
A, B – Начало и конец заданного интервала;
H – шаг с которым выбирается точка на интервале;
X – текущая точка на интервале;
Y – значение функции вычисленное в точке интервала.
Программа:

```
program primer_13;
var a,b,h,x,y:real;
begin
writeln('введите начало интервала, конец
интервала и шаг');
readln(a,b,h);
x:=a;
while x<=b do
begin
y:=sqr(x);
writeln('x=',x:5:2,'   y=',y:5:2);
x:=x+h;
```

```
end;
end.
```

Окно вывода (исходные данные введены верно, начало интервала меньше или равно концу, шаг меньше длины отрезка):

```
введите начало интервала, конец интервала и шаг
0 1 0.1
x= 0.00   y= 0.00
x= 0.10   y= 0.01
x= 0.20   y= 0.04
x= 0.30   y= 0.09
x= 0.40   y= 0.16
x= 0.50   y= 0.25
x= 0.60   y= 0.36
x= 0.70   y= 0.49
x= 0.80   y= 0.64
x= 0.90   y= 0.81
x= 1.00   y= 1.00
```

Окно вывода (исходные данные введены неверно, конец интервал – меньше чем начало):

```
введите начало интервала, конец интервала и шаг
1 0 0.1
```

Окно вывода (конец интервала совпадает с началом):

```
введите начало интервала, конец интервала и шаг
0 0 0.1
x= 0.00   y= 0.00
```

Окно вывода (расстояние между точками больше чем длина интервала)

```
введите начало интервала, конец интервала и шаг
0 0.5 0.6
x= 0.00   y= 0.00
```

Оператор repeat

Если в программе необходимо описать структуру, представленную на рис. 55, то необходимо использовать оператор цикла repeat.

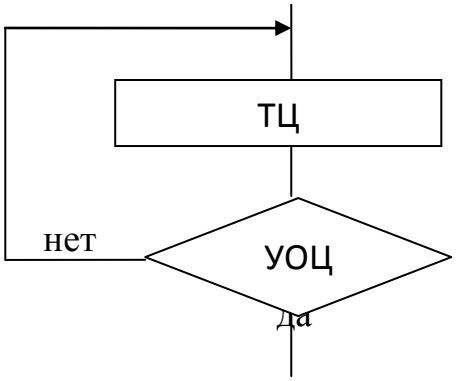


Рис. 55 Базовая алгоритмическая структура цикл с последующим условием

Общий вид оператора цикла с последующим условием:

repeat

ТЦ;

until УОЦ;

Здесь:
ТЦ – тело цикла;
УОЦ – условие окончания цикла.
Состоит оператор из:
– заголовка repeat (что означает, с этого места начинается повтор);
– тело цикла (оператор или серия операторов, которая должна многократно повторяться в программе);
– условие окончания цикла until УОЦ (условие, при выполнении которого цикл должен завершить свою работу).
Последовательность выполнения оператора:
1. Выполняются операторы тела цикла (ТЦ);
2. Вычисляется выражение УОЦ;
3. Если в результате вычисления выражения в пункте 2 имеем истину, то цикл завершается. Управление программой передается следующему после цикла оператору;
4. Если в результате вычисления выражения пункта 2 получим истину, то все циклически повторяется с пункта 1.

В этом цикле, сколько бы операторов в теле цикла не присутствовало, операторные скобки не нужны, т.к. repeat и until уже ограничивают тело цикла.

Чтобы избежать зацикливания необходимо следить, чтобы как минимум одна из величин присутствующих в условии окончания цикла меняла свое значение в теле цикла

Пример 13 а). Вычислить функцию $y = x^2$ в точках интервала $x \in [a.b]$, точки на интервале выбираются с заданным шагом h.

Обозначения:

A, B – Начало и конец заданного интервала;

H – шаг, с которым выбирается точка на интервале;

X – текущая точка на интервале;

Y – значение функции, вычисленное в точке интервала.

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 13 а). Она без пояснения приведена на рис. 54

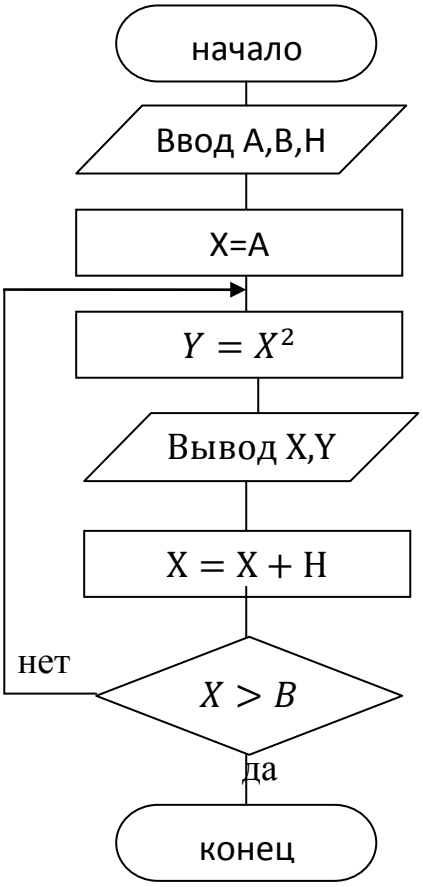


Рис 54. Блок-схема алгоритма решения задачи примера 13 а)

Программа:

```
program primer_13_A;
var a,b,h,x,y:real;
begin
writeln('введите начало интервала, конец
интервала и шаг');
readln(a,b,h);
x:=a;
repeat
y:=sqr(x);
writeln('x=',x:5:2,'   y=',y:5:2);
x:=x+h;
until x>b;
end.
```

Окно вывода (исходные данные введены верно, начало интервала меньше или равно концу, шаг меньше длины отрезка):

```
введите начало интервала, конец интервала и шаг
0 1 0.1
x= 0.00   y= 0.00
x= 0.10   y= 0.01
x= 0.20   y= 0.04
x= 0.30   y= 0.09
x= 0.40   y= 0.16
x= 0.50   y= 0.25
x= 0.60   y= 0.36
x= 0.70   y= 0.49
x= 0.80   y= 0.64
x= 0.90   y= 0.81
x= 1.00   y= 1.00
```

Окно вывода (исходные данные введены неверно, конец интервал – меньше чем начало):

```
введите начало интервала, конец интервала и шаг
1 0 0.1
x= 1.00   y= 1.00
```

Комментарий. Особенность цикла с последующим условием в том, что один раз он выполнится в любом случае

Окно вывода (конец интервала совпадает с началом)

```
введите начало интервала, конец интервала и шаг
0 0 0.1
```

```
x= 0.00    y= 0.00
```

Окно вывода (расстояние между точками больше чем длина интервала)

```
введите начало интервала, конец интервала и шаг
0 0.5 0.6
x= 0.00    y= 0.00
```

Пример 14. Рассмотрим известный алгоритм нахождения НОД (наибольшего общего делителя) двух натуральных чисел (алгоритм Евклида).

Математическая модель НОД(A,B)= НОД(min(A,B), |A-B|),

Обозначения:

A,B – заданные числа.

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 14. Блок-схема без пояснений представлена на рис. 55.

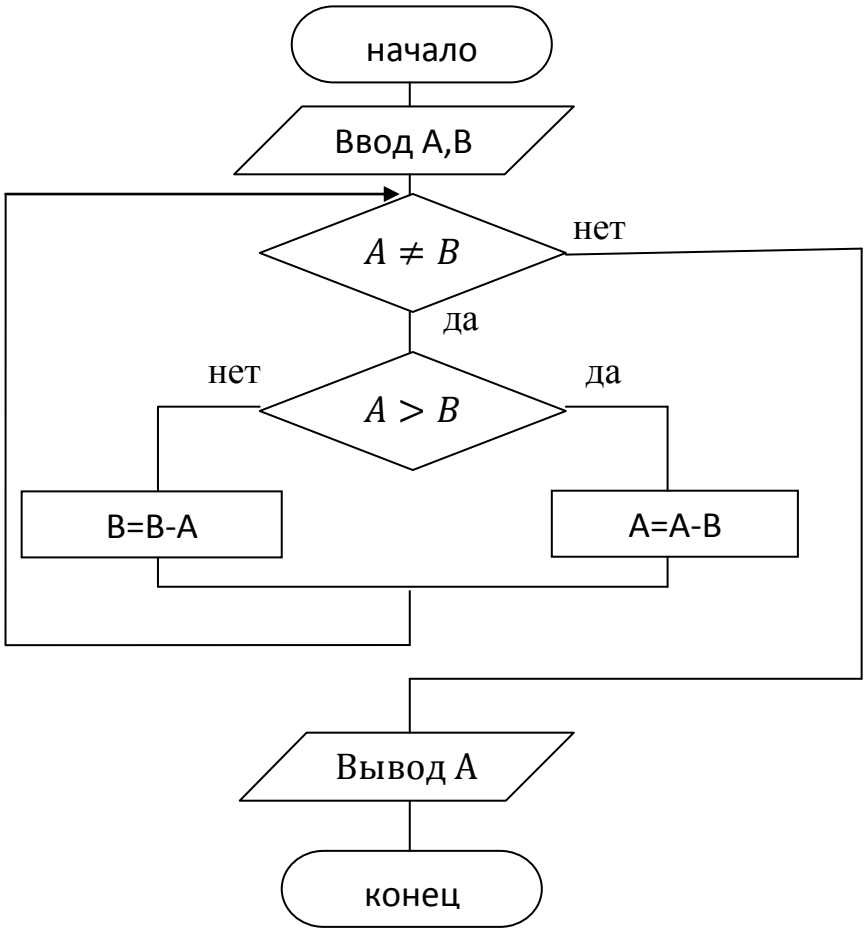


Рис. 55. Блок-схема алгоритма решения задачи примера 14

Алгоритм применим к любым натуральным числам и за конечное число шагов должен приводить к решению

Программа:

```
program primer_14;
var a,b:byte;
begin
writeln('Введите два натуральных числа');
readln(a,b);
while a<>b do
if a>b then a:=a-b else b:=b-a;
writeln('a=',a);
end.
```

Окно вывода:

```
Введите два натуральных числа
2 15
a=1
```

Пример 15. Вычислить:

$$S = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots = \sum_{i=0}^{\infty} \frac{x^n}{n!}$$

Вычислим рекуррентный множитель R:

$$R = \frac{U_{n+1}}{U_n} = \frac{X^{N+1}N!}{(N+1)!\,X^N} = \frac{x}{N+1}$$

Т.е.

$$U_{N+1} = \frac{U_n X}{N+1}$$

Обозначения:

X – любое число;

ε – точность;

N – номер слагаемого;

S – текущая сумма;

U – текущее слагаемое.

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 15. Блок-схема без пояснений представлена на рис. 56.

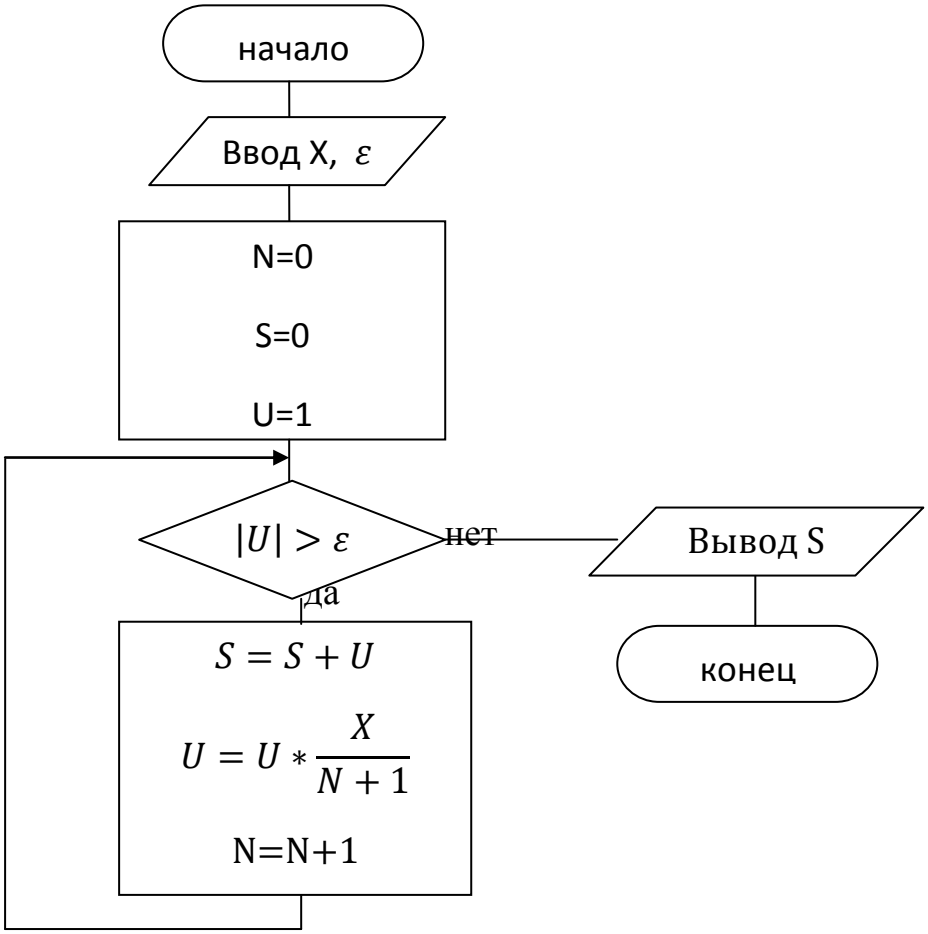


Рис. 56. Блок-схема алгоритма решения задачи примера 15

Программа:

```
program primer_15;
var x,e,s,u:real;
n:integer;
begin
writeln('Введите любое число x и точность');
readln(x,e);
n:=0;
s:=0;
u:=1;
while abs(u)>e do
begin
s:=s+u;
u:=u*x/(n+1);
```

```
n:=n+1;
end;
writeln('s=',s);
writeln('экспонента  в  точке     ',x:5:2,'
=',exp(x));
end.
```

Окно вывода:

```
Введите любое число x и точность
1 0.001
s=2.71805555555556
экспонента в точке   1.00 =2.71828182845905
```

Комментарий. Выполнена проверка вычисленного результата. В условии задачи задано разложение в ряд Тейлора экспоненты. В программу добавлен оператор

```
writeln('экспонента  в  точке     ',x:5:2,'
=',exp(x));
```

В окне вывода результат работы этого оператора

```
экспонента в точке   1.00 =2.71828182845905
```

Видим, что вычисленная с помощью нашего алгоритма сумма ряда S с точностью до 0.001 совпадает со значением, вычисленным в контрольном примере e^1.

Оператор for

Оператор цикла с параметром в паскале может быть представлен в двух видах. В этом пособии рассмотрим только один из них. Используется оператор при программировании структуры представленной на рис. 57.

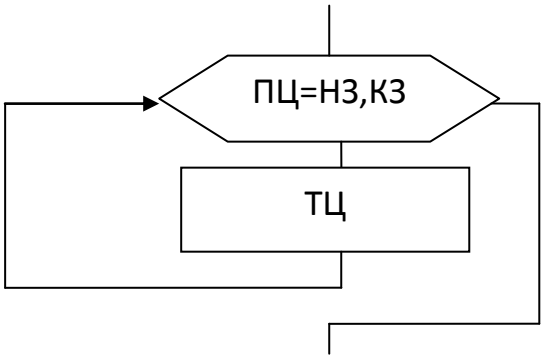


Рис 57. Схема базовой алгоритмической структуры цикл с параметром

Здесь;

ПЦ – переменная цикла или параметр цикла или счетчик повторения цикла;

НЗ – начальное значение переменной цикла;

КЗ – конечное значение переменной цикла;

ТЦ – тело цикла.

В общем виде оператор цикла с параметром будет выглядеть так:

for ПЦ:=НЗ to КЗ do

ТЦ;

Читается оператор так:

Для всех значений параметра цикла (ПЦ) изменяющегося в диапазоне от начального значения (НЗ) до конечного значения (КЗ) делать тело цикла (ТЦ).

Параметр цикла – всегда простая переменная целого типа. В этом варианте цикла при каждом новом повторе цикла параметр цикла будет увеличиваться на единицу.

Если в теле цикла будет более одного оператора, то эту серию операторов необходимо заключить в операторные скобки для того, чтобы точка с запятой после первого оператора в теле цикла не спровоцировала конец цикла.

Алгоритм выполнения оператора

При первом обращении к оператору for определяются начальные и конечные значения переменной цикла и, переменной цикла присваивается начальное значение.

После этого циклически выполняются следующие действия:

1. Проверяется условие, текущее значение переменной цикла меньше или равно конечному значению переменной цикла.

2. Если условие в пункте 1 – истина, то выполняется тело цикла, если – ложь, то цикл завершается и управление передается следующему после цикла оператору.

3. Переменная цикла увеличивается на единицу.

Пример 16. Вычислить:

$$S = \sum_{i=1}^{n} \frac{1}{i^2}$$

Обозначения:

N – Количество слагаемых в сумме;

I – номер слагаемого;

S – текущая сумма.

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 16. Блок-схема без пояснений приведена на рис. 58.

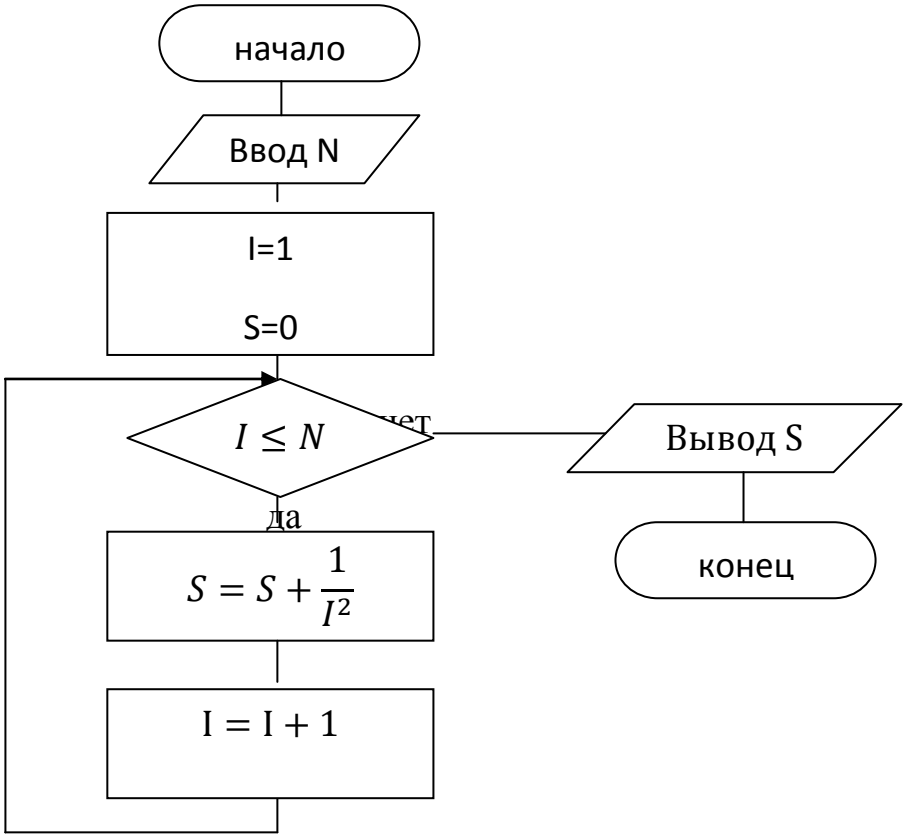


Рис. 58. Блок-схема алгоритма решения задачи примера 16

Программа:

```
program primer_16;
var s:real;
n,i:integer;
begin
writeln('Введите количество слагаемых в
сумме');
readln(n);
i:=1;
s:=0;
while i<=n do
begin
```

```
s:=s+1/sqr(i);
i:=i+1;
end;
writeln('s=',s);
end.
```

Окно вывода:

```
Введите количество слагаемых в сумме
3
s=1.36111111111111
```

Проверку производим, вычисляя сумму вручную

$$\frac{1}{1^1}+\frac{1}{2^2}+\frac{1}{3^2}=1+0{,}25+0.111111111=1{,}361111111$$

Блок-схема этого же алгоритма с использованием структуры цикл с параметром представлена на рис. 59.

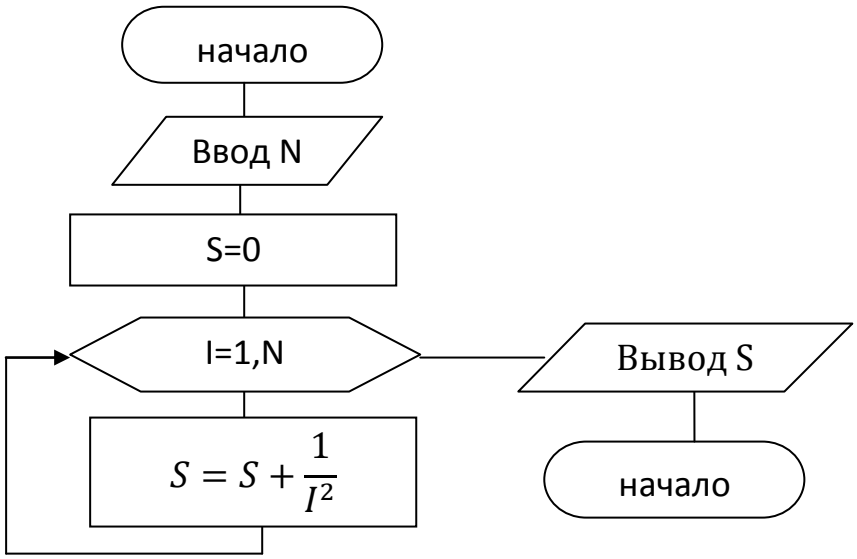


Рис. 59. Блок-схема алгоритма решения задачи примера 16

Программа:

```
program primer_16;
var s:real;
n,i:integer;
begin
writeln('Введите количество слагаемых в
сумме');
readln(n);
s:=0;
for i:=1 to n do
s:=s+1/sqr(i);
```

```
writeln('s=',s);
end.
```

Окно вывода:

```
Введите количество слагаемых в сумме
3
s=1.36111111111111
```

Комментарии. Программный код значительно короче и не перегружен дополнительными деталями. Это более подходящий выбор цикла при решении примера 16.

Циклы с переадресацией

Одномерный массив – конечная последовательность пронумерованных элементов. В описании одномерного массива задан только один индекс – номер элемента. При описании массива в паскале нужно указать три составляющих:

Имя массива.

Его структуру (одномерный, двумерный, трехмерный и т.д.).

Тип элементов массива (поскольку массив – это структурированное множество элементов одного типа).

Изобразить одномерный массив можно так

a_1	a_2	…		a_i					a_n

Если такое множество называется А, то на первом месте стоит элемент массива А с номером 1 a_1 , на втором – элемент массива А с номером 2 a_2 , на i-ом месте a_i и т.д.

Элементы массива a_i называются индексированными переменными. В паскале обращение к ним происходит так a[i].

Присваивание начальных значений (заполнение, ввод массива) заключается в присваивании каждому его элементу некоего значения заданного типа. Поскольку количество элементов массива (его размерность) должна быть известна из условия задачи, то процедура ввода массива, а также процедура вывода массива на внешний носитель наиболее эффективно будет реализована в цикле for.

Описывать массивы нужно, например, так:

Var a:array [1..10] of integer;

Читается эта запись так:

Переменная а – это множество, одномерное, максимум из 10 элементов, все элементы целого типа.

На то, что описываемый массив а является одномерным, указывает один диапазон в квадратных скобках **1..10** (это диапазон изменения индекса элементов массива). При описании массива недопустимо использование переменных при задании этого диапазона.

Если в программе массив a будет описан следующим образом

Var a:array [1..10] of integer;

то в качестве количества элементов массива может быть введено любое целое число, не превосходящее 10, под большее не выделено памяти.

Двумерный массив – это множество одномерных массивов.

a_{11}	a_{12}	…		a_{1j}					a_{1m}
…									
a_{i1}	a_{i2}			a_{ij}					a_{im}
…									
a_{n1}	a_{n1}			a_{nj}					a_{nm}

Опишем двумерный массив с именем a, состоящий максимум из 10 строк и 20 столбцов все элементы которого – вещественные числа.

Var a:array [1..10,1..20] of real;

В квадратных скобках указываем теперь два диапазона, для номеров строк и столбцов.

Пример 17. Вычислить:

$$S = \sum_{i=1}^{n} x_i y_i$$

Т.е. $x_1 y_1 + x_2 y_2 + \cdots + x_{n+1} y_{n+1}$.

Блок схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 17. Блок схема без пояснений на рис. 60.

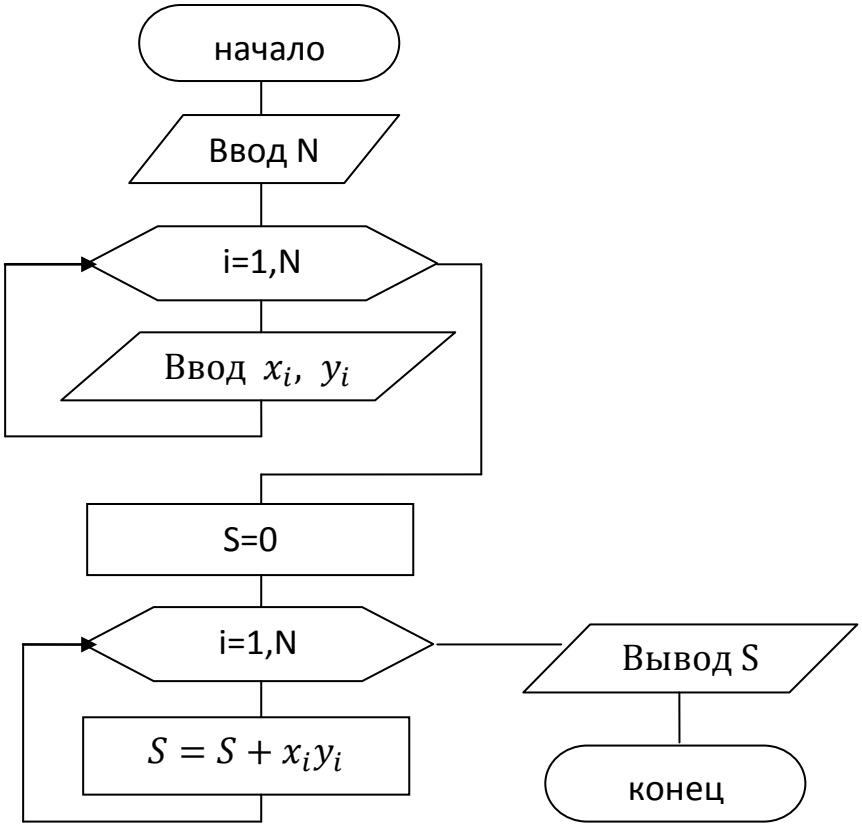


Рис. 60. Блок-схема алгоритма решения задачи примера 17

Обозначения:
X, Y – одномерные массивы;
N – количество слагаемых в сумме, и размерность массивов;
S – искомая сумма;
I – номер элемента в массиве.

Программа:

```
program primer_17;
var i,n:integer;
s:real;
x,y:array[1..50] of real;
begin
writeln('введите количество элементов массивов
x и y');
readln(n);
for i:=1 to n do
begin
writeln('введите x[',i,']   и ','   y[',i,']');
readln(x[i],y[i]);
end;
s:=0;
for i:=1 to n do
s:=s+x[i]*y[i];
writeln('s=',s);
end.
```

Окно вывода:

```
введите количество элементов массивов x и y
3
введите x[1]   и    y[1]
1 1
введите x[2]   и    y[2]
2 2
введите x[3]   и    y[3]
3 3
s=14
```

Комментарии. Был введен массив x состоящий из трех элементов со значениями

$x_1 = 1, x_2 = 2, x_3 = 3$ и

Массив y состоящий из трех элементов со значениями

$y_1 = 1, y_2 = 2, y_3 = 3.$

Результат вычислений тестируем вручную:

$$1 * 1 + 2 * 2 + 3 * 3 = 14$$

Пример 18. Вычислить количество положительных и отрицательных элементов в массиве A из N элементов

Блок схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 18. Блок схема без пояснений на рис. 61.

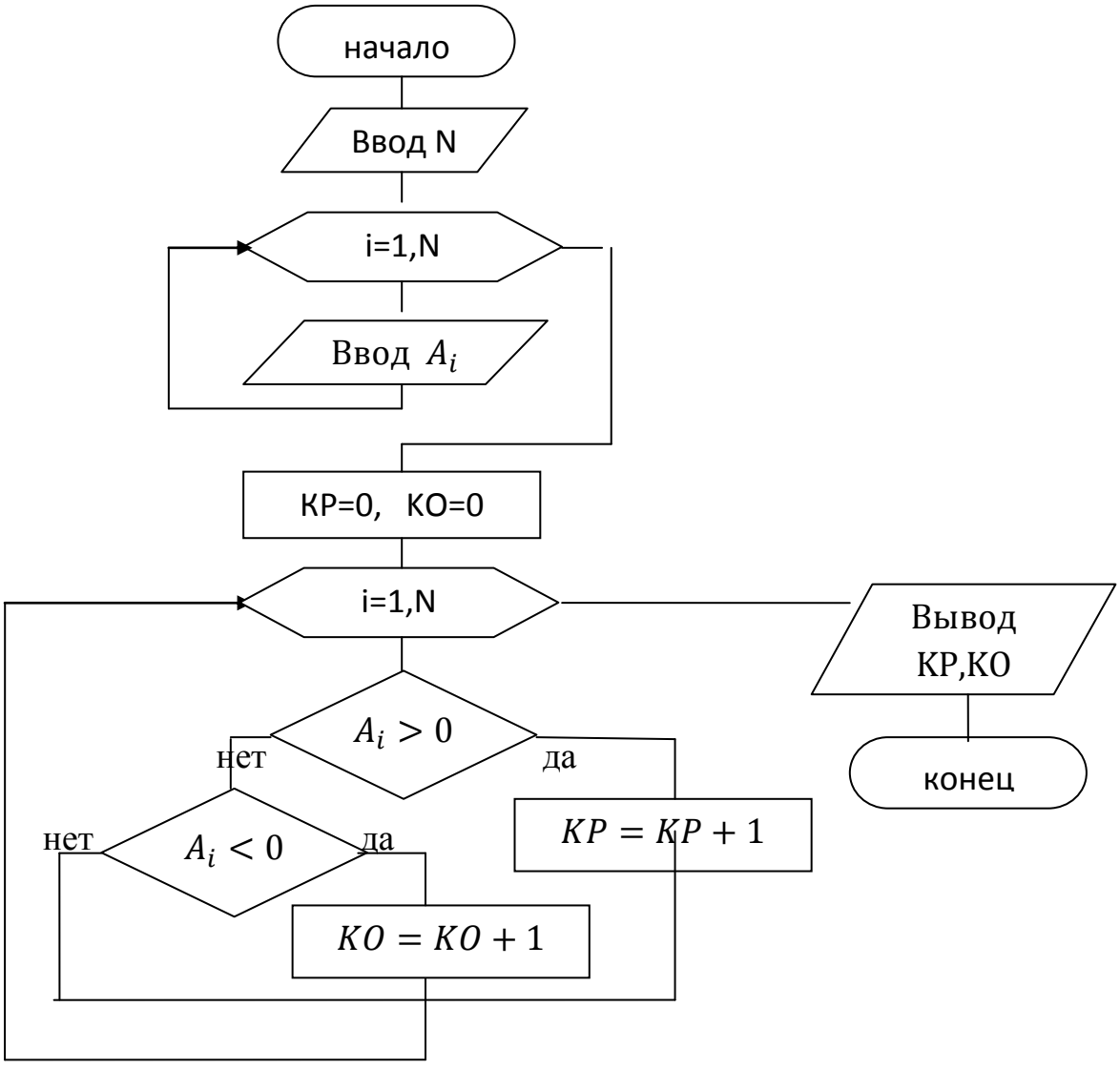


Рис. 61. Блок схема алгоритма решения задачи примера 18

Обозначения:

N – размерность массива;

i – номер элемента в массиве;

A – одномерный массив;

KP – количество положительных элементов в массиве;

KO – количество отрицательных элементов в массиве.

Программа:

```
program primer_18;
var i,n,ko,kp:integer;
a:array[1..50] of real;
begin
writeln('введите количество элементов
массива');
readln(n);
for i:=1 to n do
begin
writeln('введите a[',i,']');
readln(a[i]);
end;
ko:=0;kp:=0;
for i:=1 to n do
if a[i]>0 then kp:=kp+1 else
if a[i]<0 then ko:=ko+1;
writeln('ответ: ko=',ko,'   kp=',kp);
end.
```

Окно вывода:

```
введите количество элементов массива
5
введите a[1]
2
введите a[2]
-1
введите a[3]
0
введите a[4]
4
введите a[5]
8
ответ: ko=1   kp=3
```

Пример 19. Найти максимальный из положительных элементов одномерного массива.

Обозначения:

A – массив;

N – количество элементов в массиве;

i – номер элемента в массиве;

MAX – максимальный из положительных элементов массива.

Блок-схема алгоритма разобрана в пособии «Информатика. Основы алгоритмизации». Пример 19. Блок схема без пояснения на рис. 62.

Программа:

```
program primer_19;
var i,n:integer;
max:real;
a:array[1..50] of real;
begin
writeln('введите количество элементов
массива');
readln(n);
for i:=1 to n do
begin
writeln('введите a[',i,']');
readln(a[i]);
end;
max:=a[1];
for i:=1 to n do
if a[i]>0 then if a[i]>max then max:=a[i];
if max>0 then writeln('ответ: max=',max) else
writeln('ответ: при таких исходных данных
решения у задачи нет');
end.
```

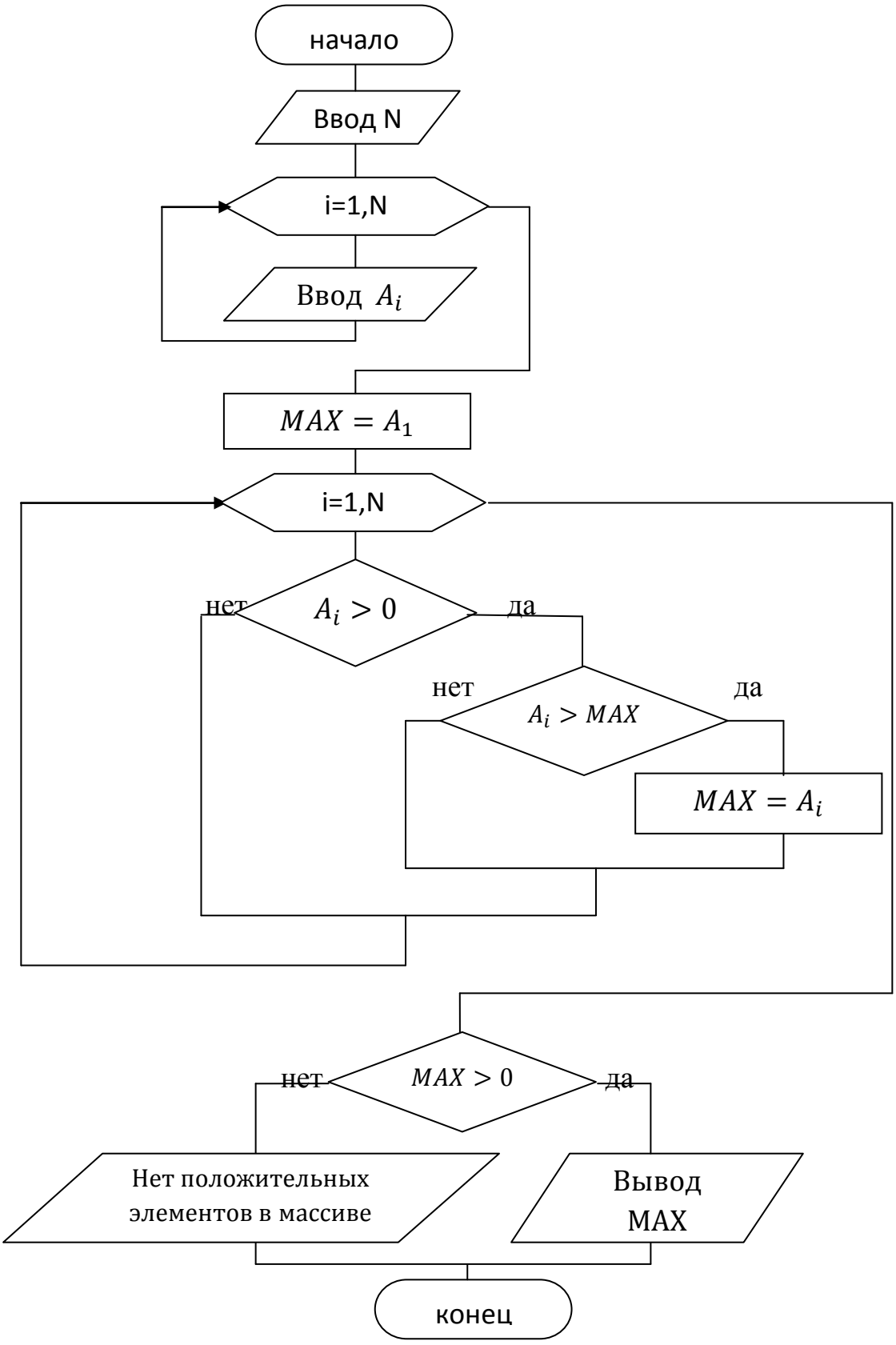


Рис. 62. Блок-схема алгоритма решения задачи примера 19

Окно вывода (вариант исходных данных, где присутствуют положительные элементы в массиве):

```
введите количество элементов массива
5
введите a[1]
2
введите a[2]
-6
введите a[3]
0
введите a[4]
3
введите a[5]
8
ответ: max=8
```

Окно вывода (вариант исходных данных, где в массиве нет положительных элементов):

```
введите количество элементов массива
5
введите a[1]
0
введите a[2]
-5
введите a[3]
-1
введите a[4]
0
введите a[5]
-44
ответ: при таких исходных данных решения у задачи нет
```

Задания для самостоятельной работы

1. Вычислить сумму натуральных чётных чисел не больших заданного натурального числа N.

2. Вычислить факториал натурального числа N.

3. Дано натуральное число N. Определить K – количество делителей этого числа, меньших чем само заданное число N (например, для N=12 делители 1, 2, 3, 4, 6. Количество K=5).

4. Дано натуральное число N. Определить, является ли оно простым. Натуральное число N называется простым, если оно делится без остатка только на единицу и на само себя. Число 19 – простое, так как делится только на 1 и 19, а число 12 не является простым, так как делится на 1, 2, 3, 4, 6 и 12.

5. Из N заданных вещественных чисел определить наибольшее число.

6. Целые числа вводятся до тех пор, пока не ввели число 0. Найти наименьшее число среди положительных. Если найденный минимум – не единственный (есть повторы), определить, сколько их.

7. Водятся числа до тех пор, пока не введет число 0. Определить среднее арифметическое всех введенных чисел.

8. Водятся числа до тех пор, пока не введет число 0. Вычислить процент положительных и отрицательных чисел.

9. Вводится последовательность из N положительных целых чисел. Найти наименьшее число среди чётных элементов последовательности.

10. Определить, является ли последовательность из N произвольных чисел строго возрастающей (каждый следующий элемент больше предыдущего).

11. Водятся числа до тех пор, пока не введет число 0. Определить, является ли эта последовательность строго убывающей (каждый следующий элемент меньше предыдущего).

12. Водятся числа до тех пор, пока не введет число 0. Вычислить среднее значение чётных элементов последовательности.

13. Водится одномерный массив из N произвольных чисел, найти среднее значение отрицательных элементов этого массива.

13. Водится одномерный массив из N произвольных чисел, Определить, содержит ли массив хотя бы два соседних одинаковых числа.

14. Водятся числа до тех пор, пока не введет число 0. Найти наибольшее число среди чисел больших, чем заданное число K

15. Водятся числа до тех пор, пока не введет число 0. Вычислиь сумму и количество отрицательных чисел среди введенных.

16. Задан массив из N действительных чисел. Найти сумму положительных и сумму отрицательных элементов этого массива.

17. Задан массив из N элементов. Вычислить, сколько раз элементы в массиве меняют знак.

18. Дано K массивов из N элементов, $N \geq 2$. Определить количество массивов, элементы которых возрастают.

19. В массиве из N вещественных чисел определить количество простых чисел.

20. Вывести на экран таблицу значений утроенных косинусов в интервале от -B до B с шагом H.

21. Ввести целое положительное число K. Вычислить сумму натуральных нечётных чисел, не превышающих это число и произведение натуральных чётных чисел, не превышающих число K.

22. Ввести целое положительное число K. Вычислить количество натуральных чисел кратных двойке и не превышающих число K.

23. Задано целое положительное число n. Определить значение выражения:

$$p = \frac{n!}{\sum_{i=1}^{n} i}$$

24. Задано целое положительное число n. Определить значение выражения:

$$p = \frac{\sum_{i=1}^{n} i^2}{n!}$$

25. Задано целое положительное число n. Определить значение выражения:

$$p = \frac{\sum_{i=1}^{n} i - 2}{(n+1)!}$$

26. Задано целое положительное число n. Определить значение выражения:

$$p = \frac{\sum_{i=1}^{n} 3^i}{(5)!}$$

27. Задано целое положительное число n. Определить значение выражения:

$$\mathrm{p} = \sum_{\mathrm{i}=0}^{10} 2^{\mathrm{i}}$$

28. Задано целое положительное число n. Определить значение выражения:

$$\mathrm{P} = \frac{\sum_{\mathrm{i}=5}^{15} \mathrm{i}}{(2\mathrm{n}+1)!}$$

29. Вводится целое число N, $(N > 1)$, Составить алгоритм вывода наименьшего целого числа K, при котором выполняется неравенство $5K > N$.

30. Вводится целое число N, $(N > 1)$. Составить алгоритм вывода наибольшего целого числа K, при котором выполняется неравенство $5K < N$.

31. Найти минимальное число большее 100, которое нацело делится на 15.

32. Найти минимальное из трех заданных чисел.

33. Найти произведения двух наименьших чисел из трех заданных.

Список литературы

1. Д. Прайс. Программирование на языке Паскаль. Практическое руководство. –М.: Мир, 1987 год. 232 стр..

2. М.А. Черкасов. Практический курс программирования на ПАСКАЛЕ. Уч. пособие. –М.: Изд-во МАИ. 2005. 186 с.

3. Е.Р. Алексеев, О.В. Чеснокова, Т.В. Кучер. Free Pascal и Lazarusю Учебник по программированию. Донецк.Ж ДонНТУ, Технопарк ДОнНТУ УНИТЕХ, 2009 . -503 с.

4. А. Епанешников, В. Епанешников. Программирование в среде Turbo Pascal 7.0. Москва; Диалог МИФИ. - 1995. – 288 с.

5. Т.А. Павловская. Паскаль. Программирование на языке высокого уровня. Учебник. Питер. 2010 г. 464 с.

6. В. Потапахин. TURBO PASCAL. Решение сложных задач. – СПб.: БХВ-Питербург, 2006. 208 с.

7. Минакова Н.И., Невская Е.С., Угольницкий Г.А. и др. Методы программирования. М.: Вузовская книга, 1999. 280 с. (2-е издание: М.: Вузовская книга, 2000).

8. Фаронов. Turbo Pascal 7.0 –СПб.: БХВ-Петербург, 2004. - 1056 с.

Оглавление

ВВЕДЕНИЕ 3

Классификационные признаки паскаля 4

Этапы написания программы 4

Структура программа на языке Паскаль 5

Синтаксис языка 6

Типы данных Паскаля 7

Операции Паскаля 8

Логические операции. 10

Некоторые стандартные функции 10

Ввод – вывод данных 11

Операторы паскаля..... 14
Оператор присваивания..... 14
Задания для самостоятельной работы..... 17
Условные операторы..... 18
Задания для самостоятельной работы..... 39
Циклический алгоритм..... 45
Оператор while..... 45
Оператор repeat..... 48
Оператор for..... 55
Циклы с переадресацией..... 59
Задания для самостоятельной работы..... 66
Список литературы..... 68

Учебное издание

Волобуева Татьяна Витальевна

Информатика

Часть 2

Методическое пособие
Для студентов бакалавриата, обучающихся по направлениям 130301 профиля “Теплоэнергетика и теплотехника”, 210301 профиля ”Нефтегазовое дело”

Подписано в печать 2009. Формат 60×84 1/16. Уч.-изд. л.
Усл. – печ. л. Бумага писчая. Тираж экз. Заказ №

Отпечатано: отдел оперативной полиграфии
Воронежского государственного технического университета
394006, Воронеж, ул.20-летия Октября, 84