**МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ УЗБЕКИСТАН**

**САМАРКАНДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ АЛИШЕРА НАВОИ**

ПРОГРАММИРОВАНИЕ. ОСНОВЫ РАБОТ В СРЕДЕ Pascal ABC

Методические указания

Рекомендовано к опубликованию учебно-методическим советом Самаркандского государственного университета (13 июня 2015 год, протокол № 8)

САМАРКАНД – 2016

УДК: 681.14
ББК: 73
П-784

Программирование. Основы работ в среде Pascal ABC. Методические указания. – Самарканд: Изд-во СамГУ, 2016. – 72 стр.

Данное методическое указание предназначено для изучения основных понятий программирование по курсу Информатика для бакалавров, обучающихся по всем направлениям факультета естественных наук. Данная дисциплина изучается на первом курсе. Приведено описание основ работы в среде Pascal ABC, рассмотрены примеры выполнения заданий в данной среде по основным темам раздела «Программирование на языке Паскаль». В конце каждой темы приведены примеры для практических занятий и самостоятельных работ, а для самостоятельного контроля – вопросы и тесты.

Составители:

канд. физ.-мат. наук, доц. **А.АБДИРАШИДОВ,**
ассистент **Б.Б.АМИНОВ,**
магистрант **А.А.АБДУРАШИДОВ.**

Ответственный редактор

канд. физ.-мат. наук, доц. **С.АМРИДИНОВ.**

Рецензенты:

канд. физ.-мат. наук, доц. **Э.Ч.ХОЛИЯРОВ,**
канд. физ.-мат. наук, доц. **Н.Р.ЗАЙНАЛОВ.**

Введение

При изучении теоретического материала по курсу «Информатика» студенты должны выполнить практические и самостоятельные работы, которые в основном связаны с программированием на языке Паскаль.

До сих пор Паскаль заслуженно считается одним из лучших языков для начального обучения программированию. Наиболее популярным решением для персональных компьютеров в 80-е – начале 90 годов стал компилятор и интегрированная среда разработки Turbo Pascal фирмы Borland, являющийся стандартом Паскаля, но редко используемый в последнее время в связи с тем, что соответствующая операционная система MS DOS устарела. Паскаль АВС – это современная версия языка программирования Паскаль и интегрированная среда, реализующая огромные возможности этой платформы в операционной системе Windows.

В данных методических указаниях работа в среде Pascal ABC рассмотрена на примерах выполнения заданий по основным темам раздела «Программирование на языке Паскаль»:

1. Понятие алгоритма и алгоритмической системы. Визуализация алгоритмов и блок-схемы. Линейные, разветвленные и циклические алгоритмы. Логические элементы программирования.

2. Составление простейших программ на языке Паскаль. Линейные алгоритмы.

3. Программирование разветвляющихся алгоритмов, операторы IF, GOTO.

4. Выполнение циклических операций. Массивы, операторы цикла, действия с матрицами.

5. Использование стандартных алгоритмов. Составление программ с использованием подпрограмм процедур и функций.

В методических указаниях приведены пояснения по наиболее часто встречающимся ошибкам, возникающим при отладке программы в среде Pascal ABC.

Данное методическое указание позволяет усовершенствовать практические навыки по изучению алгоритмических конструкций на основе языка Pascal, подготавливает основу для дальнейшего обучения программированию.

1. Основы алгоритмизации. Алгоритм. Свойства и методы изображения алгоритмов

1.1. Основные этапы решение задач

Основным этапом решения задач на компьютере считается математическое моделирование. Под математическим моделированием понимается определение характеристики изучаемого процесса с помощью математических средств. Математическое моделирование только один из этапов в курсе информатики.

Основные этапы решения задач на компьютере:

Постановка задач. Данный этап обычно осуществляется специалистом той области, к какой относится рассматриваемая задача (техника, экономика, строительство и т.д.). На этом этапе устанавливается, правильно, ли поставлена задача, и разрабатываются все необходимые критерии для её решения. Для решения любой задачи необходимо правильно понять данные, какие данные нужны и какие необходимо получит результат.

Составление математических модели задачи. На этом этапе рассматриваемая задача выражается на языке математики, т.е. составляется её математическая модель. В результате составления математической модели создаются уравнение, система уравнений и т.д. Специалист составляющей математическую модель задачи, в зависимости от области, к какой относится задача, должен хорошо знать относящийся к этой области математический аппарат. Значит, составленная математическая модель должна сохранит значение задачи.

Выбор метода решения задачи. На этом этапе выбирается метод решения полученной математической задачи. Для этого можно использовать готовые методы.

Составление алгоритма расчетов. На этом этапе составляется алгоритм решения задачи, т. е. разрабатывается порядок необходимых команд для решения задачи. Составленные алгоритмы должны быть просты и доступны.

Составление программы на одном из языков программирования. Данный этап состоит из перевода алгоритма на один из языков в программирования, доступных компьютеру. Большое значение имеет правильность составленной программы. Исполь-

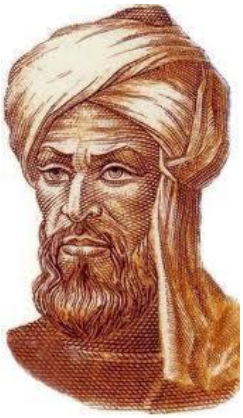
зование того или иного языка программирования, в основном завысить, от программиста.

Внесение программы в память компьютера и исправление. Для выполнения программы её нужно внести в память компьютера. На этом этапе важное значение имеет правильная работа программы, определение и исправление допущенных ошибок, исправление ошибок, допущенных при составлении алгоритма. При выполнении программы компьютером он, первую очередь переводить ее на свой «язык», так как проверяет правильно ли записана программа на и «языке» машины. Если программа записана правильно, то компьютер переходить к расчетам. В процессе расчетов тоже возможны ошибки число под корнем квадратным и т.д.

Результат. После исправления недочетов и ошибок программы компьютер используя предварительные данные переходить к выполнению программы. На этом этапе основном, выполняются расчетные работы и получается результат.

Анализ полученных результатов. Данные этом считается одним из основных этапов при потому что и произвольная программа может дать результат. Поэтому анализ правильности полученного результата, соответствия его поставленное задаче приобретает важное значение. Эта работа обычно проводится специалистом, поставившим задачу. Если полученный результат устраивает поставивший задачу, тогда можно считать, что задача на компьютере выполнено. Если результат не пригоден, то все остальные этапы решения задачи на компьютере перепроверяются. Результаты, полученные на компьютере, сопоставляются с результатами технического эксперимента или предварительного взятого точного итога.

1.2. Определение и свойства алгоритма



Определение алгоритма

Слово «Алгоритм» происходит от algorithmi - латинского написания имени аль-Хорезми, под которым в средневековой Европе знали величайшего математика из Хорезма (город в современном Узбекистане) Мухаммеда ибн Муса аль Хо-

резми, жившего в 783-850 гг. В своей книге «Об индийском счете» он сформулировал правила записи натуральных чисел с помощью арабских цифр и правила действий над ними столбиком.

Около 825 года он написал сочинение, в котором впервые дал описание придуманной в Индии позиционной десятичной системы счисления. К сожалению, арабский оригинал книги не сохранился. Аль-Хорезми сформулировал правила вычислений в новой системе и, вероятно, впервые использовал цифру 0 для обозначения пропущенной позиции в записи числа (её индийское название арабы перевели как as-sifr или просто sifr, отсюда такие слова, как «цифра» и «шифр»). Приблизительно в это же время индийские цифры начали применять и другие арабские учёные. В первой половине XII века книга аль-Хорезми в латинском переводе проникла в Европу. Переводчик, имя которого до нас не дошло, дал ей название Algoritmi de numero Indorum («Алгоритмы о счёте индийском»). По-арабски же книга именовалась Китаб альджебр валь-мукабала («Книга о сложении и вычитании»). Из оригинального названия книги происходит слово Алгебра.

В дальнейшем алгоритмом стали называть точное предписание, определяющее последовательность действий, обеспечивающую получение требуемого результата из исходных данных. Алгоритм может быть предназначен для выполнения его человеком или автоматическим устройством. Создание алгоритма, пусть даже самого простого, - процесс творческий. Он доступен исключительно живым существам, а долгое время считалось, что только человеку. Другое дело - реализация уже имеющегося алгоритма. Ее можно поручить субъекту или объекту, который не обязан вникать в существо дела, а возможно, и не способен его понять. Такой субъект или объект принято называть формальным исполнителем. Примером формального исполнителя может служить стиральная машина-автомат, которая неукоснительно исполняет предписанные ей действия, даже если вы забыли положить в нее порошок. Человек тоже может выступать в роли формального исполнителя, но в первую очередь формальными исполнителями являются различные автоматические устройства, и компьютер в том числе. Каждый алгоритм создается в расчете на вполне конкретного исполнителя. Те действия, которые может совершать исполнитель, называются его его допусти-

мыми действиями. Совокупность допустимых действий образует систему команд исполнителя. Алгоритм должен содержать только те действия, которые допустимы для данного исполнителя.

Свойства алгоритма

Данное выше определение алгоритма нельзя считать строгим - не вполне ясно, что такое «точное предписание» или «последовательность действий, обеспечивающая получение требуемого результата». Поэтому обычно формулируют несколько общих свойств алгоритмов, позволяющих отличать алгоритмы от других инструкций.

Такими свойствами являются:

- ***Дискретность*** (прерывность, раздельность) - алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов. Каждое действие, предусмотренное алгоритмом, исполняется только после того, как закончилось исполнение предыдущего.
- ***Определенность*** - каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.
- ***Допустимость*** – каждый алгоритм должен быть предназначен для определенного исполнителя, независимости от возраста и способностей.
- ***Результативность*** (конечность) - алгоритм должен приводить к решению задачи за конечное число шагов.
- ***Массовость*** - алгоритм решения задачи разрабатывается в общем виде, то есть, он должен быть применим для некоторого класса задач, различающихся только исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

Правила выполнения арифметических операций или геометрических построений представляют собой алгоритмы. При этом остается без ответа вопрос, чем же отличается понятие алгоритма от таких понятий, как «метод», «способ», «правило». Можно даже встретить утверждение, что слова «алгоритм», «способ», «правило» выражают одно и то же (т.е. являются синонимами),

хотя такое утверждение, очевидно, противоречит “свойствам алгоритма”.

Правили построения алгоритма

Само выражение «свойства алгоритма» не совсем корректно. Свойствами обладают объективно существующие реальности. Можно говорить, например, о свойствах какого-либо вещества. Алгоритм – искусственная конструкция, которую мы сооружаем для достижения своих целей. Чтобы алгоритм выполнил свое предназначение, его необходимо строить по определенным правилам. Поэтому нужно говорить все же не о свойствах алгоритма, а о правилах построения алгоритма, или о требованиях, предъявляемых к алгоритму.

Первое правило – при построении алгоритма прежде всего необходимо задать множество объектов, с которыми будет работать алгоритм. Формализованное (закодированное) представление этих объектов носит название данных. Алгоритм приступает к работе с некоторым набором данных, которые называются входными, и в результате своей работы выдает данные, которые называются выходными. Таким образом, алгоритм преобразует входные данные в выходные.

Это правило позволяет сразу отделить алгоритмы от “методов” и “способов”. Пока мы не имеем формализованных входных данных, мы не можем построить алгоритм.

Второе правило – для работы алгоритма требуется память. В памяти размещаются входные данные, с которыми алгоритм начинает работать, промежуточные данные и выходные данные, которые являются результатом работы алгоритма. Память является дискретной, т.е. состоящей из отдельных ячеек. Поименованная ячейка памяти носит название переменной. В теории алгоритмов размеры памяти не ограничиваются, т. е. считается, что мы можем предоставить алгоритму любой необходимый для работы объем памяти.

Третье правило – дискретность. Алгоритм строится из отдельных шагов (действий, операций, команд). Множество шагов, из которых составлен алгоритм, конечно.

Четвертое правило – детерминированность. После каждого шага необходимо указывать, какой шаг выполняется следующим, либо давать команду остановки.

Пятое правило – сходимость (результативность). Алгоритм должен завершать работу после конечного числа шагов. При этом необходимо указать, что считать результатом работы алгоритма.

Итак, алгоритм – неопределяемое понятие теории алгоритмов. Алгоритм каждому определенному набору входных данных ставит в соответствие некоторый набор выходных данных, т. е. вычисляет (реализует) функцию. При рассмотрении конкретных вопросов в теории алгоритмов всегда имеется в виду какая-то конкретная модель алгоритма.

1.3. Виды алгоритмов и их реализация

Алгоритм применительно к вычислительной машине – точное предписание, т.е. набор операций и правил их чередования, при помощи которого, начиная с некоторых исходных данных, можно решить любую задачу фиксированного типа.

Виды алгоритмов как логико-математических средств отражают указанные компоненты человеческой деятельности и тенденции, а сами алгоритмы в зависимости от цели, начальных условий задачи, путей ее решения, определения действий исполнителя подразделяются следующим образом:

1. ***Механические*** алгоритмы, или иначе детерминированные, жесткие (например, алгоритм работы машины, двигателя и т.п.);

2. ***Гибкие*** алгоритмы, например, стохастические, т.е. вероятностные и эвристические.

Механический алгоритм задает определенные действия, обозначая их в единственной и достоверной последовательности, обеспечивая тем самым однозначный требуемый или искомый результат, если выполняются те условия процесса, задачи, для которых разработан алгоритм.

1. ***Вероятностный*** (стохастический) алгоритм дает программу решения задачи несколькими путями или способами, приводящими к вероятному достижению результата.

2. ***Эвристический*** алгоритм (от греческого слова “эврика”) – это такой алгоритм, в котором достижение конечного результата программы действий однозначно не предопределено, так же как не обозначена вся последовательность действий, не выявлены все действия исполнителя. К эвристическим алгоритмам отно-

сят, например, инструкции и предписания. В этих алгоритмах используются универсальные логические процедуры и способы принятия решений, основанные на аналогиях, ассоциациях и прошлом опыте решения схожих задач.

3. ***Линейный*** алгоритм – набор команд (указаний), выполняемых последовательно во времени друг за другом.

4. ***Разветвляющийся*** алгоритм – алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов.

5. ***Циклический*** алгоритм – алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными. К циклическим алгоритмам сводится большинство методов вычислений, перебора вариантов.

Цикл программы – последовательность команд (серия, тело цикла), которая может выполняться многократно (для новых исходных данных) до удовлетворения некоторого условия.

Вспомогательный (подчиненный) алгоритм (процедура) – алгоритм, ранее разработанный и целиком используемый при алгоритмизации конкретной задачи. В некоторых случаях при наличии одинаковых последовательностей указаний (команд) для различных данных с целью сокращения записи также выделяют вспомогательный алгоритм.

Методы изображения алгоритмов

На практике наиболее распространены следующие формы представления алгоритмов:

1. словесная (записи на естественном языке);
2. графическая (изображения из графических символов);
3. псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);
4. программная (тексты на языках программирования).

Словесное описание алгоритма

Данный способ получил значительно меньшее распространение из-за его многословности и отсутствия наглядности.

Рассмотрим пример на алгоритме нахождение максимального из двух значений:

1. Определим форматы переменных X, Y, M, где X и Y – значения для сравнения, M – переменная для хранения максимального значения;
2. получим два значения чисел X и Y для сравнения;
3. сравним X и Y;
4. если X меньше Y, значит большее число Y;
5. поместим в переменную M значение Y;
6. если X не меньше (больше) Y, значит большее число X;
7. поместим в переменную M значение X.

Словесный способ не имеет широкого распространения по следующим причинам: такие описания строго не формализуемы; страдают многословностью записей; допускают неоднозначность толкования отдельных предписаний.

1.4. Блок-схема алгоритма

А этот способ оказался очень удобным средством изображения алгоритмов и получил широкое распространение в научной и учебной литературе.

Структурная (блок-, граф-) схема алгоритма – графическое изображение алгоритма в виде схемы связанных между собой с помощью стрелок (линий перехода) блоков – графических символов, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия.

Графическое изображение алгоритма широко используется перед программированием задачи вследствие его наглядности, т.к. зрительное восприятие обычно облегчает процесс написания программы, ее корректировки при возможных ошибках, осмысливание процесса обработки информации.

Можно встретить даже такое утверждение: «Внешне алгоритм представляет собой схему – набор прямоугольников и других символов, внутри которых записывается, что вычисляется, что вводится в машину и что выдается на печать и другие средства отображения информации «Здесь форма представления алгоритма смешивается с самим алгоритмом.

Принцип программирования «сверху вниз» требует, чтобы блок-схема поэтапно конкретизировалась и каждый блок «расписывался» до элементарных операций. Но такой подход можно осуществить при решении несложных задач. При решении

сколько-нибудь серьезной задачи блок-схема «расползется» до такой степени, что ее невозможно будет охватить одним взглядом.

Блок-схемы алгоритмов удобно использовать для объяснения работы уже готового алгоритма, при этом в качестве блоков берутся действительно блоки алгоритма, работа которых не требует пояснений. Блок-схема алгоритма должна служить для упрощения изображения алгоритма, а не для усложнения.

Блок «процесс» применяется для обозначения действия или последовательности действий, изменяющих значение, форму представления или размещения данных. Для улучшения наглядности схемы несколько отдельных блоков обработки можно объединять в один блок. Представление отдельных операций достаточно свободно.

Блок «решение» используется для обозначения переходов управления по условию. Блок «модификация» используется для организации циклических конструкций. (Слово модификация означает видоизменение, преобразование). Внутри блока записывается параметр цикла, для которого указываются его начальное значение, граничное условие и шаг изменения значения параметра для каждого повторения.

Блок «предопределенный процесс» используется для указания обращений к вспомогательным алгоритмам, существующим автономно в виде некоторых самостоятельных модулей, и для обращений к библиотечным подпрограммам.

В таблице приведены часто употребляемые символы.

Название символа	Обозначение и пример заполнения	Пояснение
Процесс	x=(a-b)/sin(l)	Вычислительное действие или последовательность действий
Решение	да a<b нет	Проверка условий
Модификация	i=1,50,2	Начало цикла
Предопределенный процесс	Расчет параметров	Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод	Ввод a,b,c	Ввод-вывод в общем виде
Пуск-останов	Начало	Начало, конец алгоритма, вход и выход в подпрограмму
Документ	Печать a,c	Вывод результатов на печать

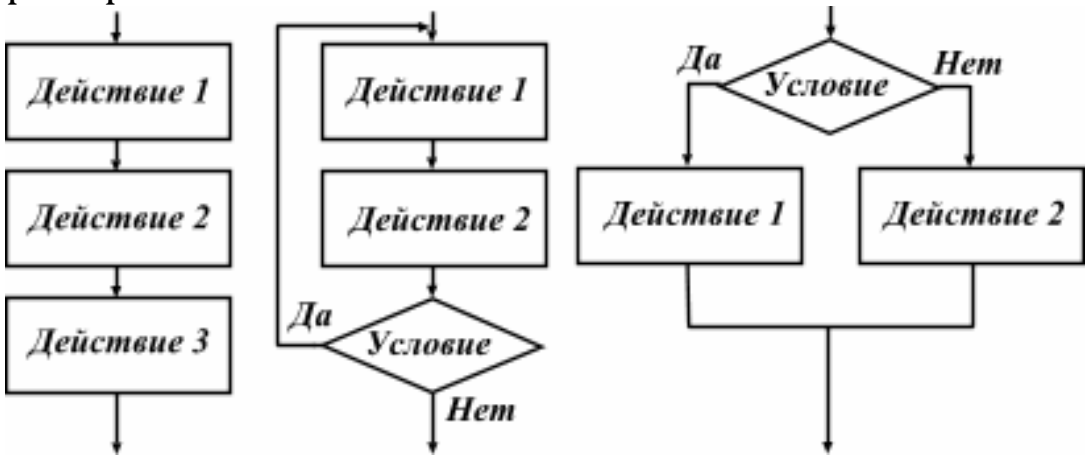
Программное представление алгоритма

При записи алгоритма в словесной форме, в виде блок-схемы или на псевдокоде допускается определенный произвол при изображении команд. Вместе с тем такая запись точна настолько, что позволяет человеку понять суть дела и исполнить алгоритм.

Однако на практике в качестве исполнителей алгоритмов используются специальные автоматы — компьютеры. Поэтому алгоритм, предназначенный для исполнения на компьютере, должен быть записан на «понятном» ему языке. И здесь на первый план выдвигается необходимость точной записи команд, не оставляющей места для произвольного толкования их исполнителем.

Следовательно, язык для записи алгоритмов должен быть формализован. Такой язык принято называть языком программирования, а запись алгоритма на этом языке — программой для компьютера.

Примеры:



Контрольные вопросы

1. Что вы понимаете под математическим моделированием?
2. Назовите основные этапы решение задач на компьютере.
3. Каково значение слова «Алгоритмы»?
4. Какие свойства имеет алгоритм?
5. Проведите методы изображения алгоритмов
6. Какие виды алгоритмов имеются?
7. Дайте характеристику линейным алгоритмам.
8. Дайте характеристику разветвляющимся алгоритмам.
9. Дайте характеристику циклическим алгоритмам.
10. Дайте определение алгоритма.
11. Назовите основные структуры алгоритмов.

12. Перечислите элементы графического представления алгоритмов.
13. Перечислите возможные виды циклов.
14. Назовите основные характеристики алгоритмов.
15. Какие особенности алгоритма Вы знаете?
16. Укажите отличие инструкции от алгоритма.

2. Работа в среде Pascal ABC

Pascal ABC — это:

- **современный язык программирования**, основанный на Delphi (Object Pascal) и сочетающий простоту языка Паскаль и огромные возможности платформы.
- бесплатная, **простая и мощная среда разработки**, ориентированная на обучение программированию.
- уникальная Web-среда, позволяющая разрабатывать и запускать программы на языке Паскаль из окна браузера, а также иметь личный каталог программ на сервере.

Скачать бесплатно последнюю версию **Pascal ABC** и ознакомиться с возможностями интегрированной среды можно на сайте **http://pascalabc.net**. При завершении записи версии на рабочем столе появится иконка, с помощью которой загружается интегрированная среда Pascal ABC и вызывается окно браузера (рис. 1).

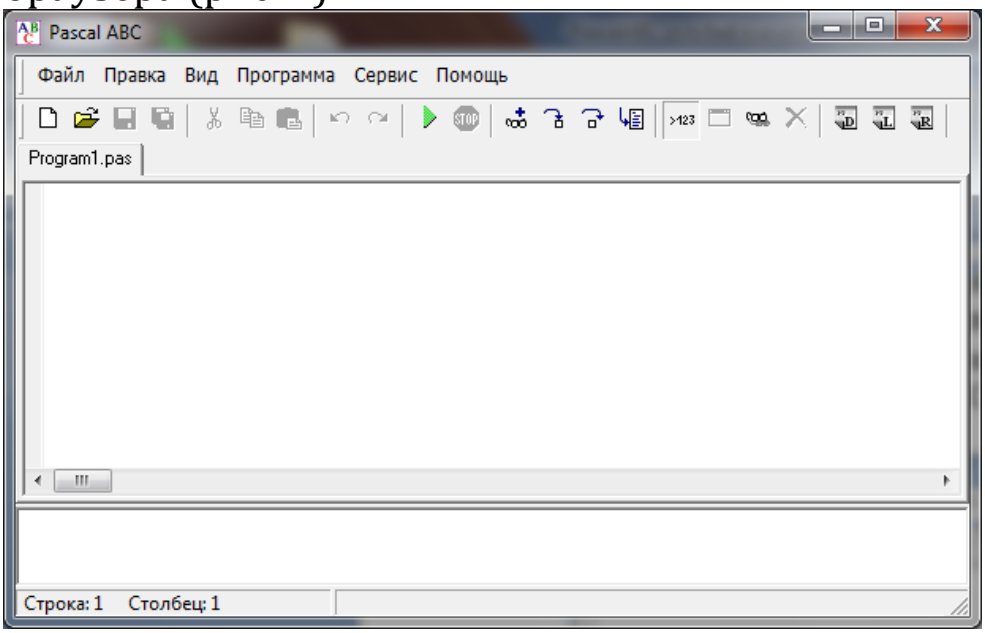
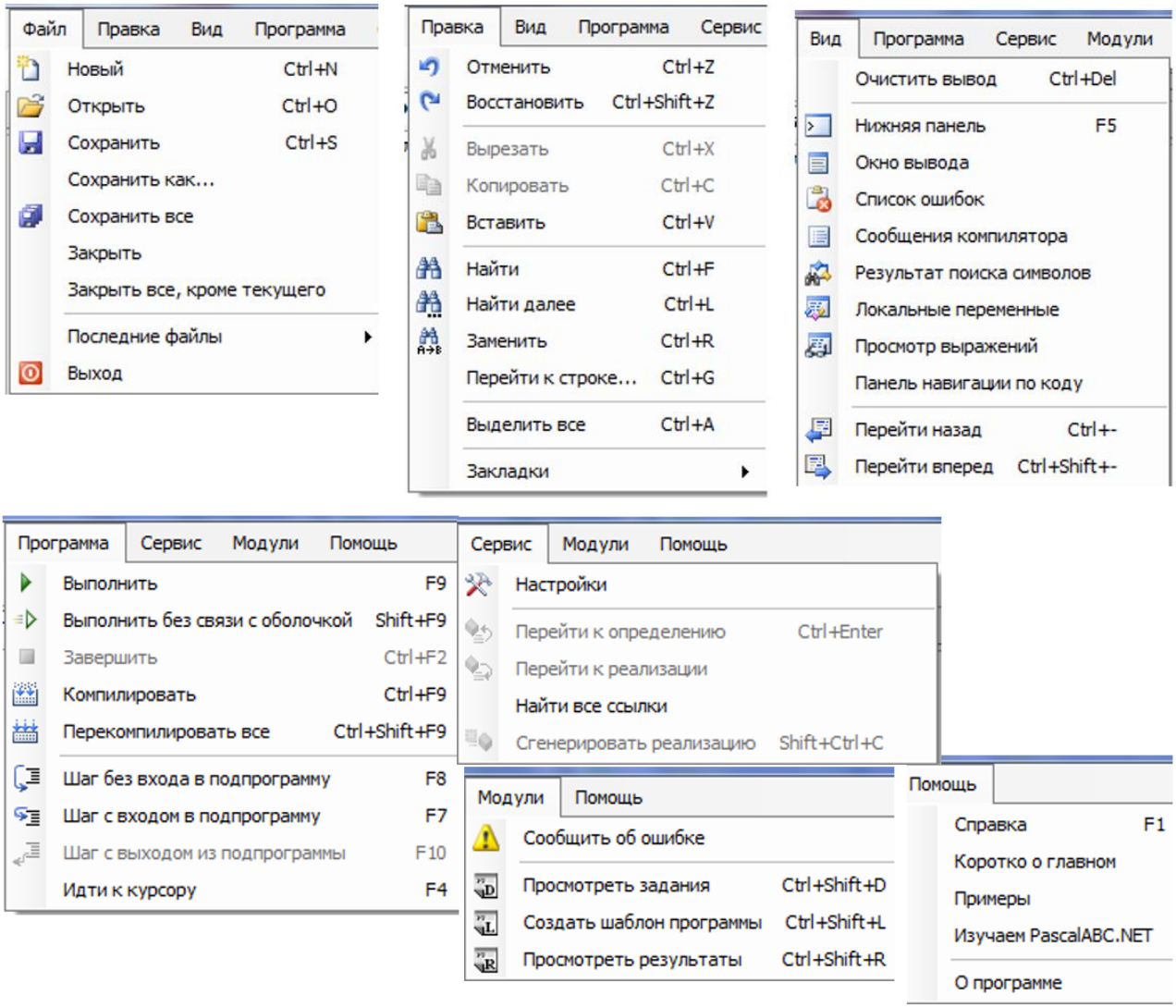


Рис. 1

При активизации элементов горизонтального меню в верхней части браузера, открываются горизонтальные подменю, обеспечивающие выполнение ряда нижеперечисленных функций:



На панель управления вынесены иконки, с помощью которых выполняются наиболее часто используемые операции:

 – новый

 – открыть

 – сохранить

 – сохранить все

 – вырезать

 – перейти вперед

 – выполнить

 – завершить

 – компилировать

 – шаг без входа в подпрограмму

 – копировать

 – вставить

 – отменить

 – восстановить

 – перейти назад

 – шаг с входом в подпрограмму

 – окно вывода

 – просмотреть задания

 – создать шаблон программы

 – просмотреть результаты

В верхнем окне браузера набирается текст программы на языке программирования Паскаль, в нижнем Окне вывода появляется результат выполнения программы. Последовательность работы в среде Pascal ABC рассмотрим на примерах выполнения заданий по основным темам раздела «Программирование на языке Паскаль».

3. Составление простейших программ на языке Паскаль. Линейные алгоритмы

Цель занятия: Получить представление о структуре программы на языке Паскаль, научиться использовать стандартные функции языка Паскаль для записи математических выражений. Освоение структуры программы, основных типов данных, операторов ввода и вывода; составление линейных алгоритмов, алгоритмов с использованием условного оператора.

Структура программы на языке Pascal

Program <имя программы>;	Заголовок программы
Label <список меток>; Const <имя константы>=<значение>; Type <имя типа>=<определение типа>; Var <имя переменной>:<тип>; Procedure <описание процедур>; Function <описание функций>;	Описательная часть
Begin <оператор 1>; <оператор 2>; ... <оператор N>; End.	Исполнительная часть (тело программы)

Элементы программы, которые в готовом коде будут заменяться пользовательскими значениями, обозначены угловыми скобками <…>.

Наиболее часто используемые типы данных.

Integer - целый тип. Значением переменной этого типа может быть целое число из отрезка [−32 768; 32 767].

Real - вещественный тип. Значением переменной этого типа может быть вещественное число, модуль которого принадлежит отрезку [2,9 ×10-39 ;1,7 ×1038].

Boolean - логический тип. Переменная этого типа может принимать только два значения: истина (true) или ложь (false).

Char - символьный тип. Значением переменной этого типа может быть любой символ из набора ASCII-символов.

Оператор присваивания «:=». Слева от оператора записывается имя переменной, которой присваивается значение, а справа - выражение, значение которого вычисляется перед присваиванием:

<имя переменной>:=<выражение>;

Пример: y:=a*x+b;

Оператор ввода позволяет указанным переменным присвоить значения, вводимые с клавиатуры:

Readln(<список переменных>);

Пример: Readln(x); или Readln(x,y);

Оператор вывода используется для вывода на экран:

- пояснений: Writeln('<Комментарий>');

Пример: Writeln('Введите x');

- значений переменных в бесформатном виде:

Writeln(<имя переменной>);

Пример: Writeln(y);

- значений переменных в форматированном виде:

Writeln(<имя переменной>:<ширина поля вывода>:<число цифр после запятой>);

Пример: Writeln(y:7:4);

- допускается также одновременный вывод пояснений и значний.

Пример: Writeln('При x=',x:5:1,' y=',y:7:4);

Составной оператор — конструкция языка программирования, состоящая из нескольких операторов, заключенных в операторные скобки

Begin ... End;

но участвующая в программе в качестве единого оператора.

Арифметические операции: сложение (+), вычитание (–), умножение (*), деление (/), деление нацело (div), остаток от деления нацело (mod).

Пример: A div B (если A=10 и B=3, то результат равен 3);

C mod D (если C=10 и D=3, то результат равен 1).

Операции отношения: равно (=), меньше (<), больше (>), менше или равно (<=), больше или равно (>=), не равно (<>).

Стандартные функции

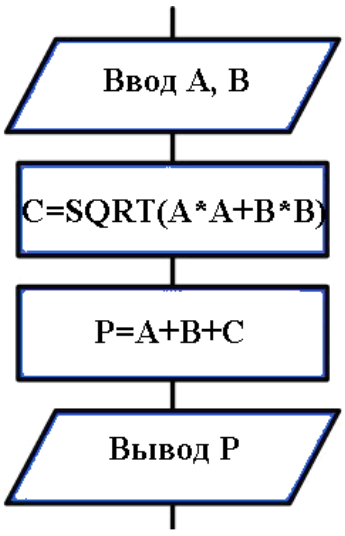
Математич. форма записи	Запись формулы в Pascal	Математич. форма записи	Запись формулы в Pascal
$\lvert x \rvert$	`abs(x)`	$[x]$	`trunc(x)`
x^2	`sqr(x)`	π	`pi`
$\sqrt{x}$	`sqrt(x)`	$\ln x$	`ln(x)`
$\sin x$	`sin(x)`	e^x	`exp(x)`
$\cos x$	`cos(x)`	x^a	`exp(a*ln(x))` при x>0
$\mathrm{arctg}\, x$	`arctan(x)`		

Простейшие задачи имеют **линейный алгоритм** решения (имеют структуру **"следование"**).

Алгоритм линейной структуры представляет собой последовательность действий и не содержит каких-либо **условий**

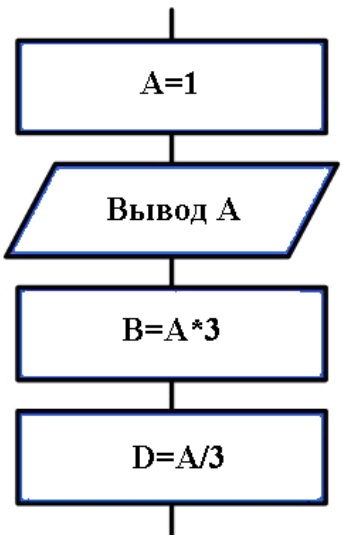
Таким образом, в таких алгоритмах все этапы решения задачи выполняются строго последовательно, т.е. линейные алгоритмы выполняются в естественном порядке его написания и не содержит разветвлений и повторений.

На практике линейные алгоритмы в чистом виде встречаются редко: при расчете арифметических и алгебраических выражений, при расчете по формулам, при решении ряда бытовых задач.



ПРИМЕР :

```
readln(A,B);
C:=sqrt(A*A+B*B);
P:=A+B+C;
writeln(P);
```



ПРИМЕР:

```
A=1;
writeln(A);
B:=A*3;
D:=A/3;
```

Пример №1. Составить программу на языке Паскаль и вычислить значение y:

$$y = a^x + \frac{ln|b| + 3{,}4 \cdot e^{a \cdot x}}{tg^2 c \cdot \sqrt{|b|}},$$

где a=0,3; b=-3.7; c=0,84; x=-5,4

Последовательность работы

1. Прежде чем приступить к программированию, выражение правой части уравнения нужно представить в виде последовательности вычислительных операций, ограниченных скобками, с использованием формул преобразований и стандартных функций языка Паскаль:

$$y = exp(x * ln(a) + (ln(abs(b) + 3.4 * exp(a * x))/(sqr(sin(c)/cos(c)) * sqrt(abs(b)))$$

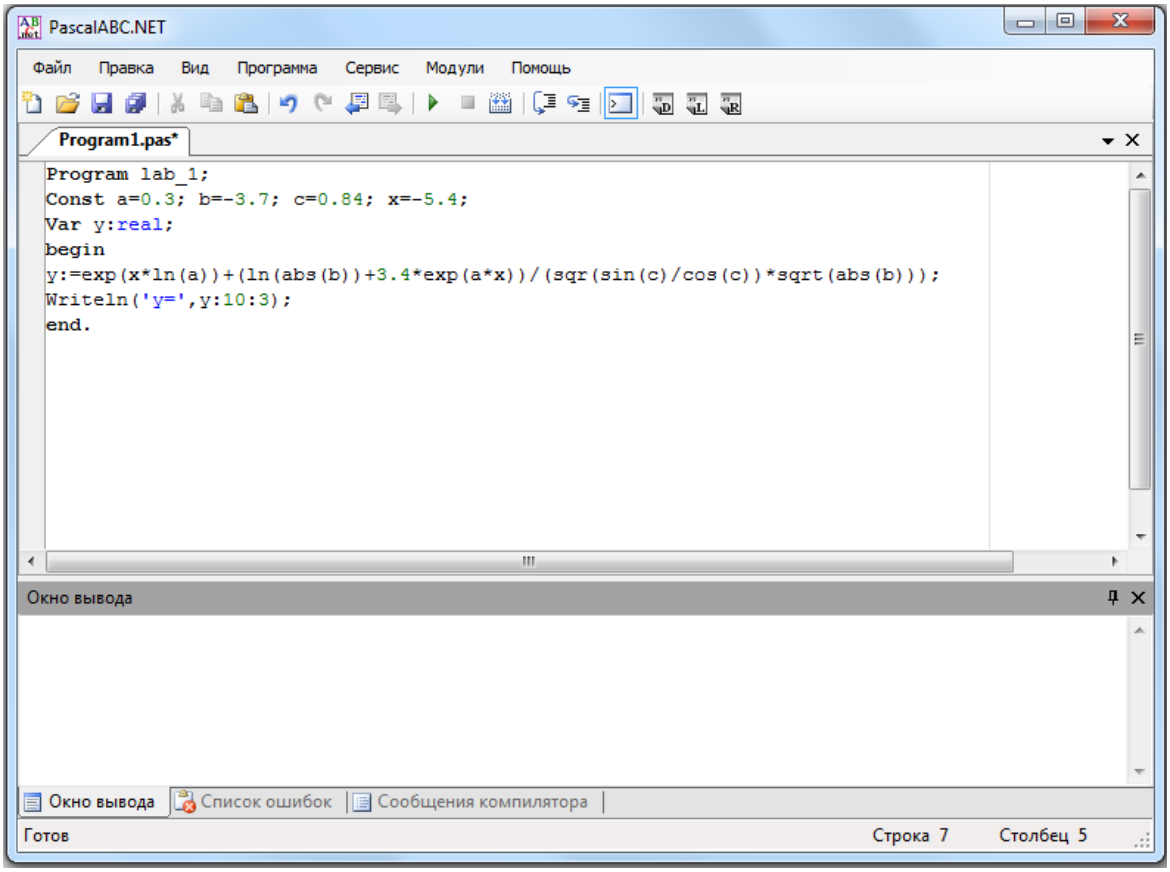
2. Составляем программу на языке Паскаль, задавая исходные данные в разделе описания констант (const):

```
Program Rab_1;
Const a=0.3; b=-3.7; c=0.84; x=-5.4;
Var y : real;
 begin
    y:=exp(x*ln(a))+(ln(abs(b))+3.4*exp(a*x))/
       (sqr(sin(c)/cos(c))*sqrt(abs(b)));
```

```
    Writeln('y=',y:10:3);
end.
```

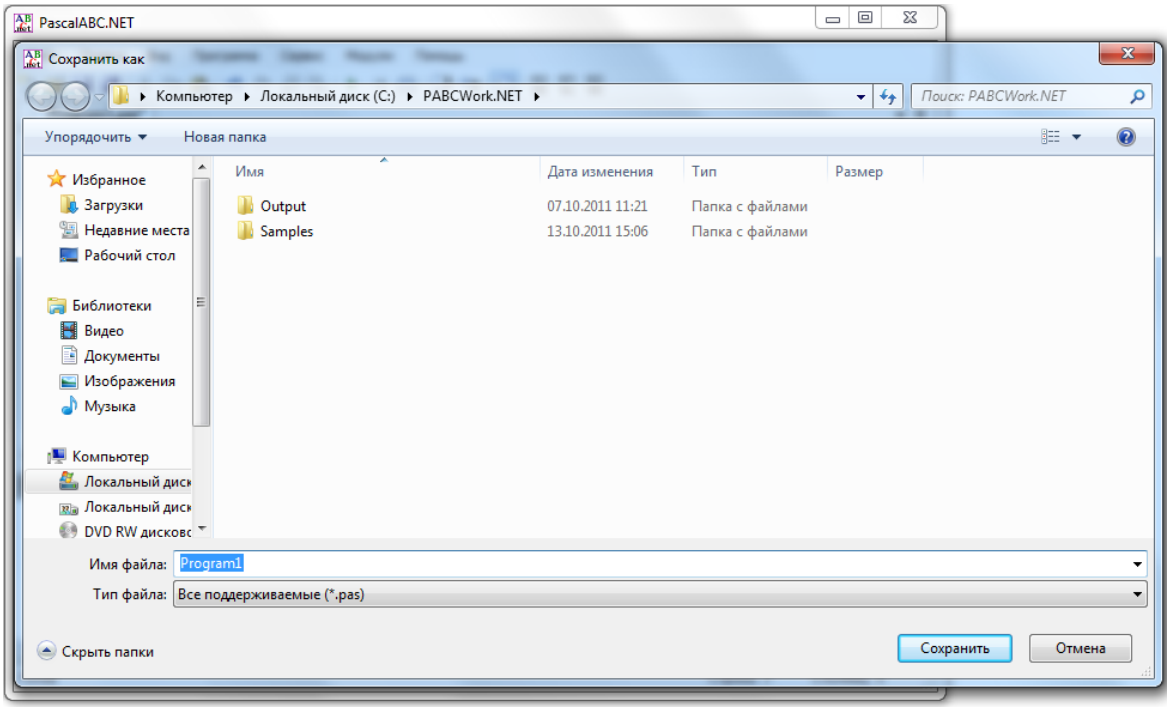
3. Входим в среду Pascal ABC, используя иконку на рабочем столе компьютера

4. В верхнем окне браузера набирается текст программы на языке Паскаль:

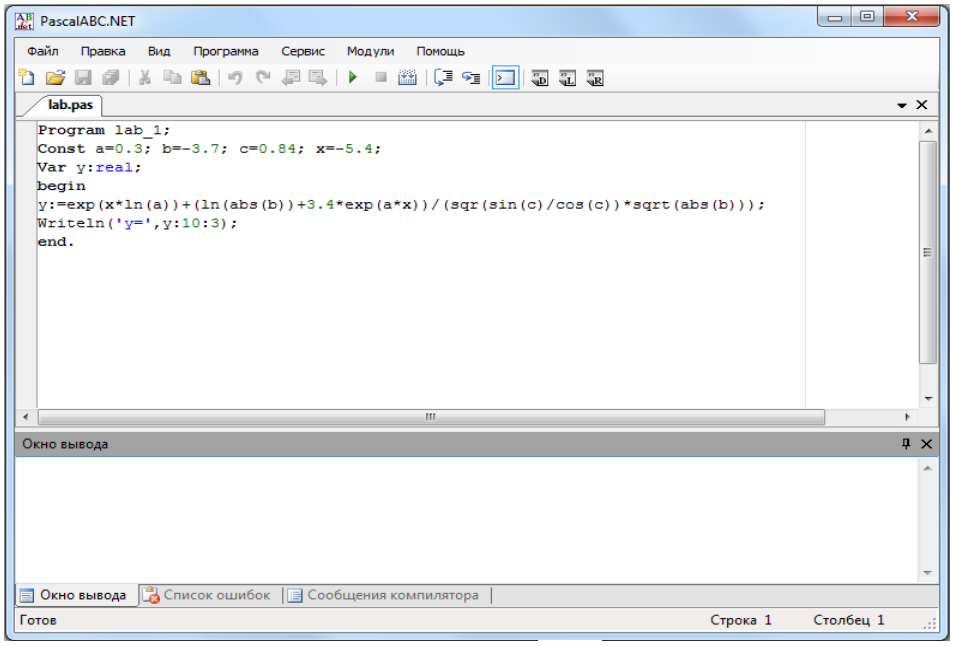


5. Необходимо сохранить текст программы, для этого воспользуемся иконкой 

Появится окно, в поле Имя файла которого вместо  следует написать имя, под которым программа будет храниться в каталоге, например lab (имя составляется из букв латинского алфавита, цифр и некоторых допустимых символов, не должно превышать 8 позиций):



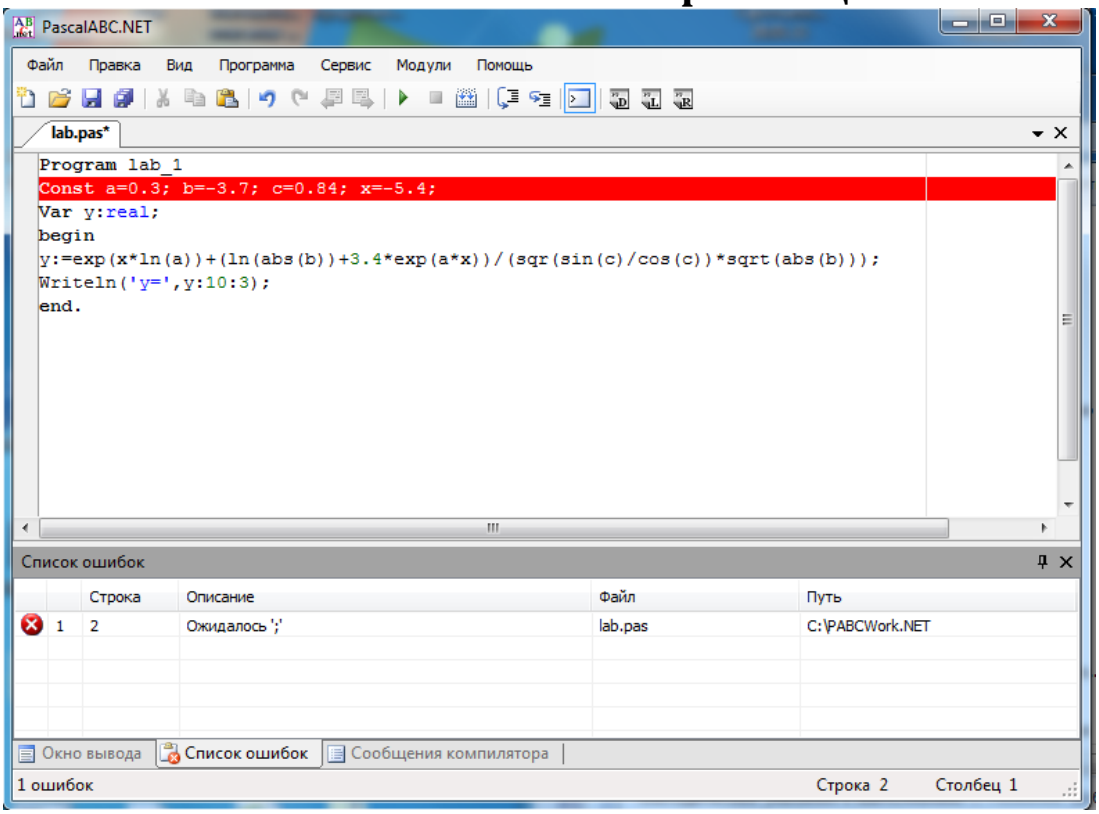
После этого указанное имя появится на вкладке рабочего окна.



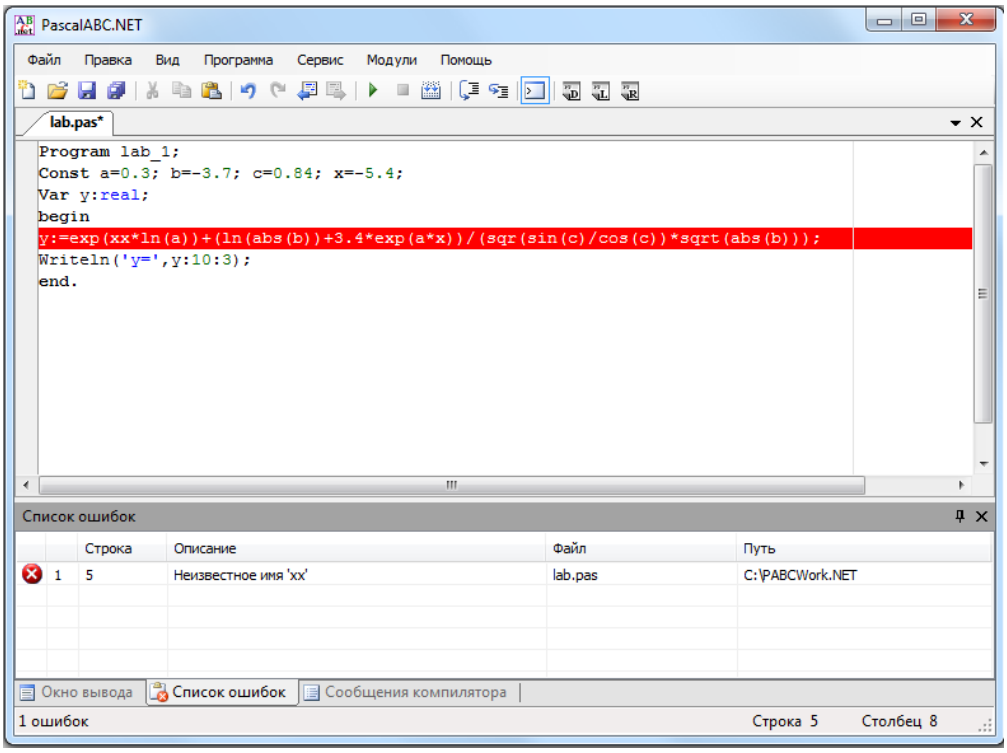
6. Воспользовавшись иконкой , выполняем программу.

7. При возникновении ошибок, в Окне вывода будет появляться подсказка. Следует отлаживать программу (исправлять ошибки), пока в Окне вывода не появится результат. Каждый раз после исправления ошибки следует сохранять последнюю версию, воспользовавшись иконкой и выполнять программу, активизируя иконку

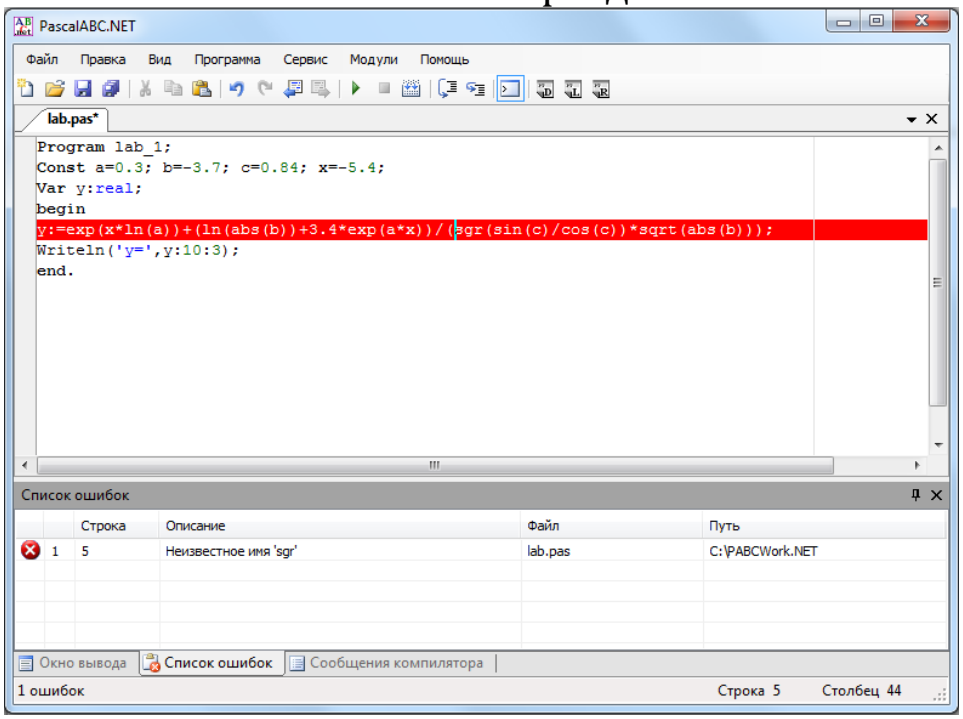
Разъяснения по наиболее часто встречающимся ошибкам



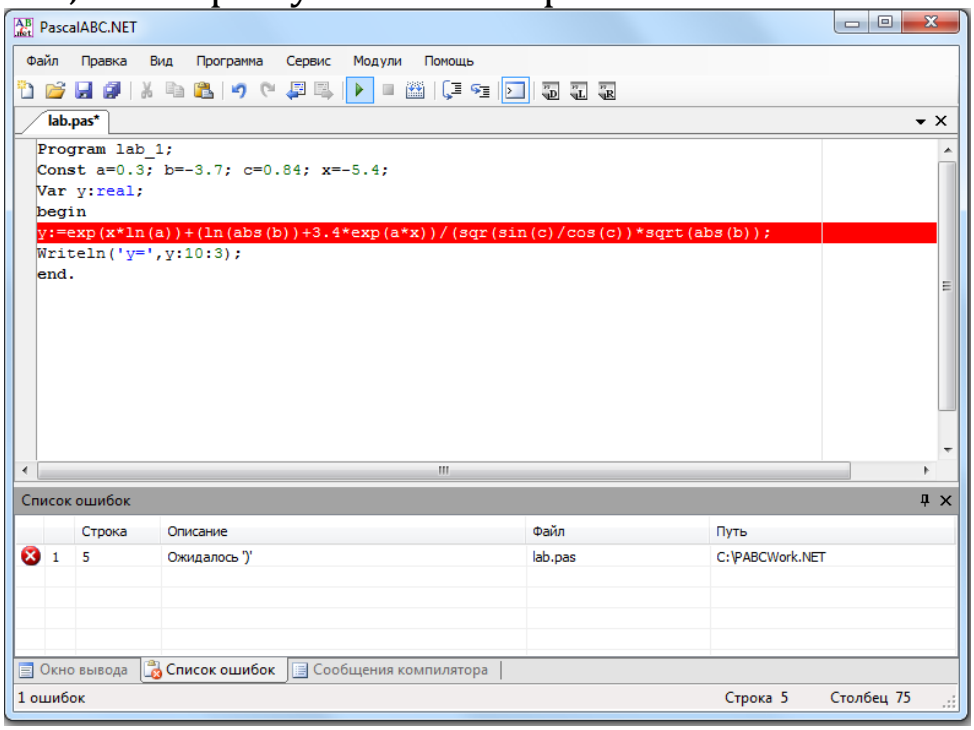
После заголовка программы **Program** lab_1 должен стоять символ ‘;’. Аналогичная ошибка может возникать при отсутствии символа ‘;’ после любого другого оператора программы. Красной строкой выделен оператор, перед которым отсутствует разделитель ‘;’, в конце предыдущего оператора следует этот символ поставить.



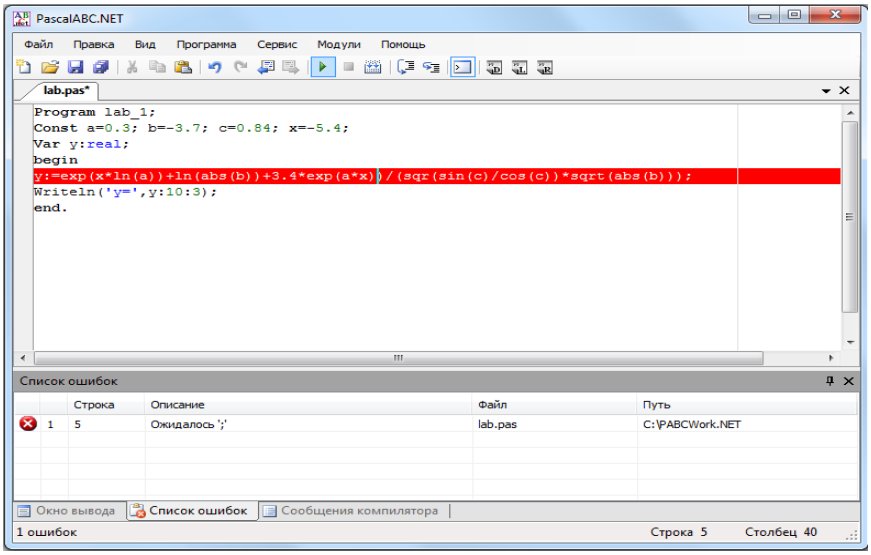
Имя ‘xx’ не описано ни в разделе описания констант (**Const**), ни в разделе описания переменных (**Var**). В данном случае, в разделе **Const** описана переменная ‘x’ (x=-5.4;), а в операторе вычисления ‘y’ используется нигде не описанная величина ‘xx’. Следует исправить ‘xx’ на ‘x’. Данная ошибка может возникать, как в случае неверного написания имени в теле программы, так и в случае отсутствия описания этого имени в разделе описаний.



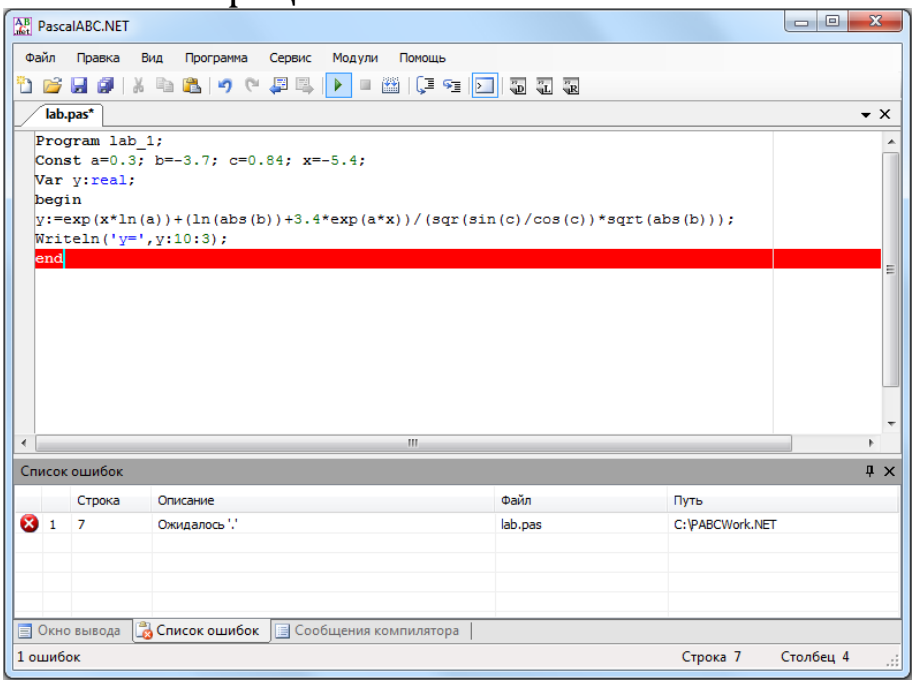
Аналогичная ошибка может возникнуть при неверном написании имени стандартной процедуры. Красной строкой выделяется строка, в которой указано неверное имя.



В данном случае, не хватает закрывающей скобки ‘)’. Это указано в нижнем окне браузера. Оператор, в который нужно внести исправления, выделен в верхнем окне красной строкой. Необходимо внимательно проверить написание оператора и указать скобку в соответствии с последовательностью выполнения вычислительных операций.

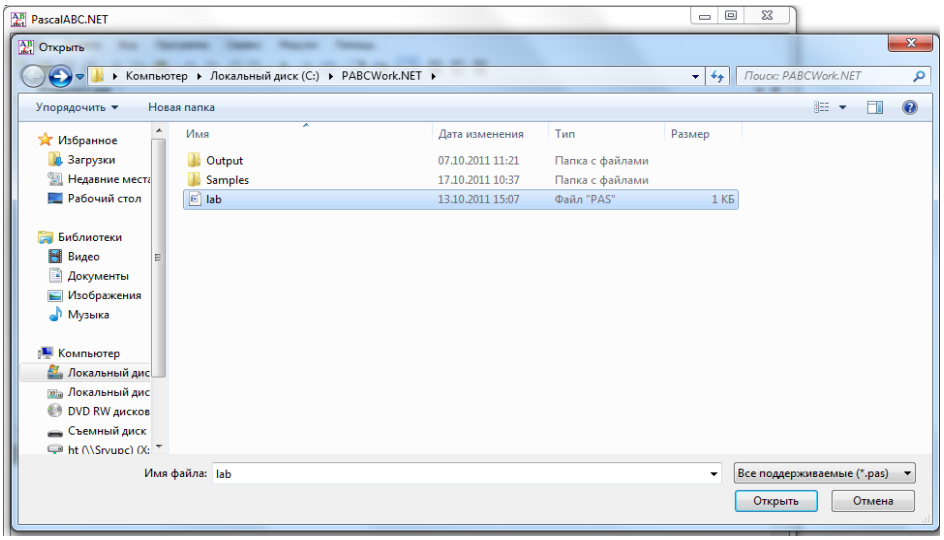


Возникновение данной ошибки связано с недостаточным количеством открывающих скобок ‘)’. Следует внимательно проверить написание оператора, выделенного красной строкой, и указать скобку в соответствии с последовательностью выполнения вычислительных операций.



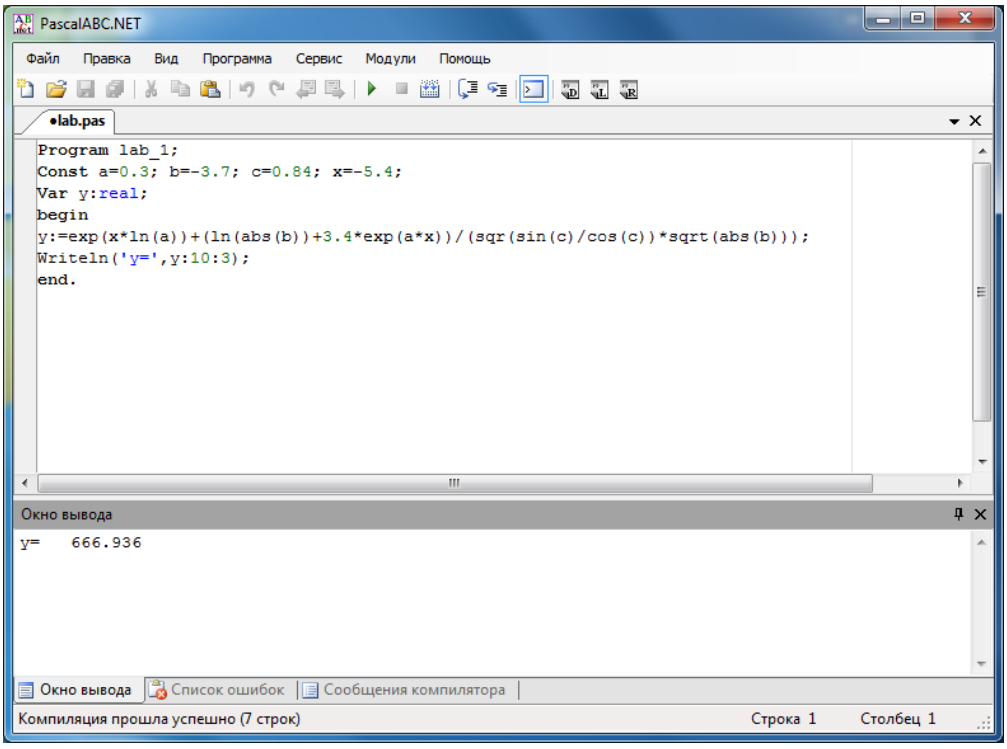
Отсутствует символ ‘.’ В конце программы.

8. Если не удалось завершить отладку программы за один прием, следует записать последнюю версию на диск, используя иконку  , и в следующий раз продолжить отладку, предварительно войдя в среду Pascal ABC и воспользовавшись иконкой  . При этом появится окно:



Следует выделить в каталоге имя отлаживаемого файла (lab) и активизировать клавишу **Открыть**, а затем продолжить отладку.

Когда программа будет отлажена и в окне вывода появится результат:



После получения результата можно приступать к оформлению отчета.

Задания для практического занятия

Задание № 1-П. Составить алгоритм, блок-схему, программу на языке Pascal для вычисления значения функции $y = f(x)$, где a = 2,5; b = 1,2 при заданном значении x = 3,6 , которое вводится с клавиатуры. Результат вывести с тремя знаками после запятой.

1. $y = \sqrt{\cos x^2 + \sin^2(x^2 + 2x + 1)}$
2. $y = a\ln\left(\sqrt{1+x} + \cos^2 x\right)$
3. $y = e^{\sqrt{1+x}} - \ln|ctgx| + 0.75$
4. $z = 5x + 3x^2\sqrt{1 - x^3 \sin 2x}(2x^2 + 9)$
5. $y = e^{\sin\sqrt{x}} + \sqrt{1 + x^2}$
6. $y = |x - 5cd| + e^{\sqrt{\sin x + 1}}$
7. $y = \dfrac{1 - x\ln|x|}{e^x}, x = \dfrac{(u+v)^2}{\dfrac{3}{4} + u^2 + v^2}$
8. $y = \sin^2 x/(x^2 + z^2)$
9. $y = \dfrac{x^3 + e^x - \sin^2 x}{\sqrt{\cos x} + |x|}$
10. $y = \dfrac{1}{\cos x} + \ln\left|tg\dfrac{x}{2}\right| + \dfrac{a + dx}{c + dx}$
11. $y = \dfrac{\sin x + a}{\cos^2 x + b^2}, a = \dfrac{\sqrt{x} + e^x}{\ln|x - 3|}$
12. $z = \dfrac{\ln|x| + y^2}{2arctgx} - n * tgx$
13. $s = \dfrac{bc\sin a}{2}, b = \sqrt{\dfrac{1 - \cos a}{2}},$ $c = \sqrt{\dfrac{1 + \cos a}{2}}$
14. $S = p(p - a)tg\dfrac{A}{2}, p = (A + B + C)/2$ $, A = \sqrt{\dfrac{1 - \cos A}{1 + \cos B}}$
15. $y = 2\sqrt{x^2 + \cos x} + \dfrac{e^{x+3}}{2}$
16. $y = \dfrac{\ln(|x| + 2)}{x^4 + 1} + e^{x+1}$
17. $y = x\cos x + \sin^2 x - \ln(|x| + 1)$
18. $u = (1 + z) * \dfrac{x + \dfrac{y}{z}}{a - \dfrac{1}{1 - x^2}}, z = \dfrac{\sin^2 x}{x^2 + y^2}$
19. $w = \dfrac{1}{a^2\sqrt{a^2 - 1}},$ $a = \dfrac{u^3 + e^4 - \sin^3 x}{e^x}\sqrt{\cos x} + |x|.$
20. $y = \sqrt{a^2 + e^2} + 4\sqrt{a/b}, a = \dfrac{\sqrt{x} + e^x}{\ln|x - 3|}$
21. $y = \sqrt{\dfrac{\sqrt{a^2 + b^2 + c^2}}{a^2 + b^2} - \dfrac{1}{2a} * e^{\frac{|x-a|}{b}}}$ $y = \dfrac{\sqrt{|\ln\|x + 1| + d}}{\sin a + \sin d}, \quad a = \sqrt{\dfrac{3vh}{nr}}, \quad d = \dfrac{1}{3}nr^2h.$
22. $s = p^2 tg\dfrac{A}{2} * tg\dfrac{B}{2} * tg\dfrac{C}{2}, \quad p = (A + B + C)/2.$
23. $y = e^{\sin\sqrt{x}} + \sqrt{1 + x^2}$; $z = \sin^2 x/(x^2 + y^2)$.
24. $y = \dfrac{\sin x + a}{\cos^2 x + b^2}, a = \dfrac{\sqrt{x} + e^x}{\ln|x - 3|}$. 25. $y = \sqrt{\dfrac{\sqrt{a^2 + b^2 + c^2}}{a^2 + b^2} - \dfrac{1}{2a} * e^{\frac{|x-a|}{b}}}$

Задания для самостоятельной работы

Задание № 1-С. Составить алгоритм, блок-схему, программу на языке Pascal для вычисления значения функции $y = f(x)$ при заданном значении x, которое вводится с клавиатуры. Результат вывести с тремя знаками после запятой. Варианты заданий приведены в таблице.

Вариант	$y = f(x)$	Исходные данные	
		const	x
1	$y = \dfrac{\sqrt{cx + 62{,}7e^x}}{ax^2 + 7x + b\ln x}$	$a = 7{,}2$ $b = 14{,}3$ $c = 13{,}4$	$x = 5{,}6$
2	$y = \dfrac{ax + 3{,}8tgx}{\sqrt{bx^3 + c}}$	$a = 1{,}23$ $b = 5{,}14$ $c = 3{,}97$	$x = 7{,}1$
3	$y = \left(\dfrac{a}{bx^2 + 1} + cx^3 + b\sin^2 x\right)^2$	$a = 2{,}27$ $b = 1{,}18$ $c = 3{,}92$	$x = 0{,}78$
4	$y = \left(a\sqrt{4{,}19x^3 - 1} - \sqrt{b\ln x + c}\right)^{-1}$	$a = 9{,}2$ $b = 3{,}5$ $c = 12{,}3$	$x = 3{,}2$
5	$y = \ln\lvert a\sin x + b\cos(x^2)\rvert$	$a = 1{,}2$ $b = 2{,}3$	$x = 5{,}6$
6	$y = \sqrt{\dfrac{ax^3 + \operatorname{arctg} x}{cx + b\lvert\ln x\rvert}}$	$a = 2{,}71$ $b = 1{,}63$ $c = 0{,}81$	$x = 0{,}51$
7	$y = \dfrac{ax}{\sqrt{b^2 + 2\,\mathrm{e}^x - bx}}$	$a = 6{,}32$ $b = 3{,}704$	$x = 7{,}15$
8	$y = \cos(ax) + b\ln(1 + bx + \mathrm{e}^x)$	$a = 7{,}1$ $b = 1{,}8$	$x = 0{,}9$
9	$y = \dfrac{\sqrt{\mathrm{e}^{ax} + x^2}\cdot\ln(x^2 + bx + 10)}{\sin(cx) + 4{,}2}$	$a = 5{,}7$ $b = 6{,}4$ $c = 3{,}1$	$x = 2{,}8$
10	$y = \dfrac{\sqrt{\mathrm{e}^{2x+b}} - 1{,}7\cos(cx)}{\ln(x^2 + a)} + x^3$	$a = 2{,}1$ $b = 5{,}3$ $c = 1{,}4$	$x = -1{,}2$
11	$y = \dfrac{\ln\sqrt{x^2 + b} + cx^3}{\mathrm{e}^x + a}$	$a = 4{,}7$ $b = 7{,}21$ $c = 1{,}72$	$x = -0{,}91$
12	$y = \dfrac{\sin\sqrt{\mathrm{e}^x + ax^2 + b\ln x}}{ax^2 + cx + 13{,}7}$	$a = 3{,}7$ $b = 4{,}9$ $c = 2{,}5$	$x = 2{,}5$

4. Программирование разветвляющихся алгоритмов

Цель занятия: Научиться составлять программы на языке Паскаль на основе разветвляющихся алгоритмов с использованием условного оператора *if…then*. Освоение структуры программы, основных типов данных, операторов ввода и вывода; составление линейных алгоритмов, алгоритмов с использованием условного оператора.

В таких алгоритмах делается выбор: выполнять или не выполнять какую-нибудь группу команд в зависимости от условия, т.е. выбирается один из нескольких возможных путей (вариантов) вычислительного процесса. Каждый подобный путь называется ветвью алгоритма.

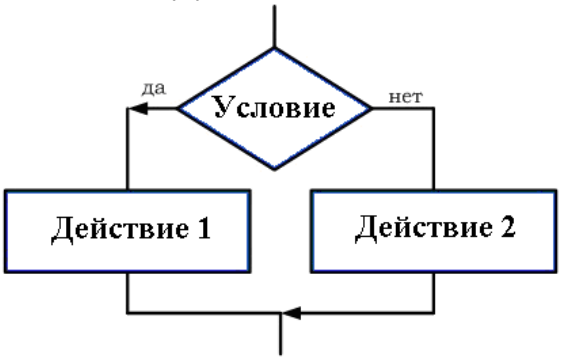
Признаком разветвляющегося алгоритма является наличие операций условного перехода, когда происходит проверка истинности некоторого логического выражения (проверяемое условие) и в зависимости от истинности или ложности проверяемого условия для выполнения выбирается та или иная ветвь алгоритма.

В логических выражениях используется операция сравнения: < (меньше), > (больше), <= (меньше или равно), >= (больше или равно), = (равно), <> (не равно). Часто встречаются задачи, в которых используются не отдельные условия, а совокупность связанных между собой условий (отношений). Для связи используются AND и (или) OR. Например: (2+3) and (2+5) >= 6 – нет (ложно)

Алгоритм предполагает выполнение **Действия 1**, если записанное условие истинно (выполняется), и выполнение **Действия 2**, если условие ложно (не выполняется) – это полная развилка.

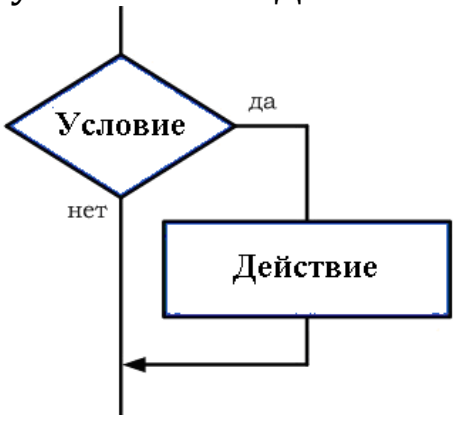
Полная развилка:

If условие **then** Действие 1 **else** Действие 2



Неполная развилка:

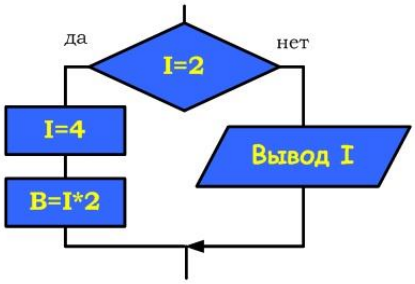
If условие **then** Действие 1



Если в алгоритме отсутствует Действие 2, т.е. если записанное

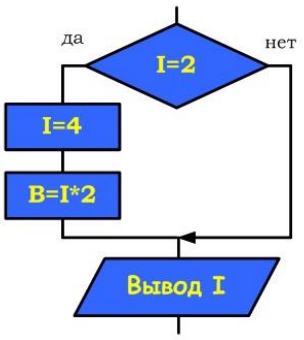
условие истинно, то выполняется Действие 1, а если условие ложно, то никаких действий не выполняется – это не полная развилка.

ПРИМЕР:




```
IF I=2 THEN
  BEGIN
        I:=4;
        B:=I*2;
  END
ELSE
WRITELN(I);
```

ПРИМЕР:




```
IF I=2 THEN
  BEGIN
        I:=4;
        B:=I*2;
  END;
WRITELN(I);
```

Пример №2. Вычислить значение:

$$y=\begin{cases} a+bx, & \text{если } x<1, \\ \sqrt{\left|a+\dfrac{b}{x}\right|}, & \text{если } x=1, \\ (ax+bx)^2, & \text{если } x>1, \end{cases}$$

$$\text{где } a=\frac{3{,}5b}{b+c^3};\ b=2{,}4;\ c=1{,}7;$$

Значение *x* задать с клавиатуры.

Последовательность работы

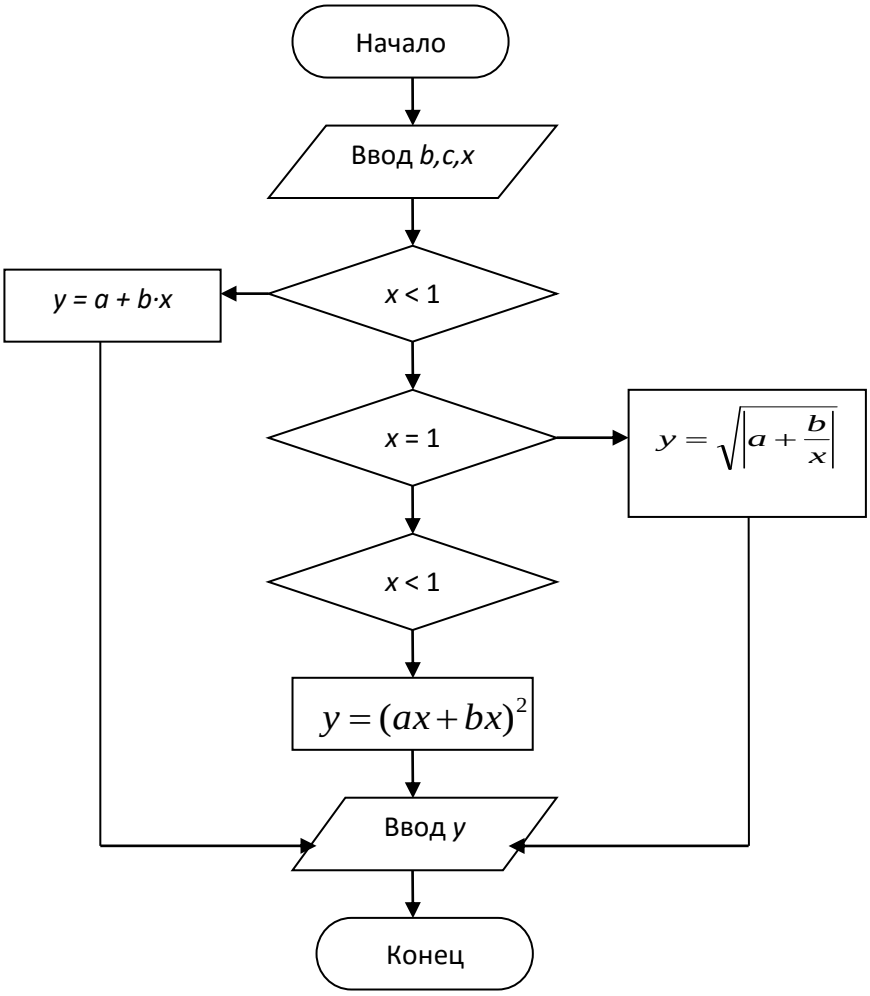
1. Составляем блок-схему алгоритма, которая будет отражать последовательность расчета и написания программы. В данном случае используем разветвляющийся тип алгоритма.

2. Составляем программу расчета на языке Паскаль, используя, для передачи управления, условный оператор перехода *if…then…else*.

Текст программы:

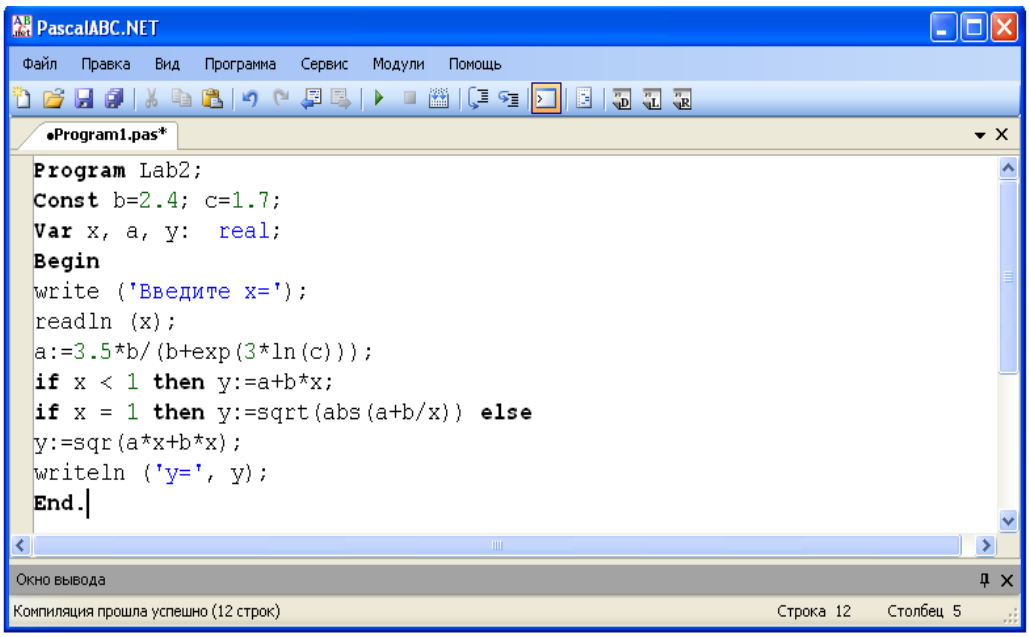
```
Program Lab2;
Const b=2.4; c=1.7;   (описание постоянных параметров)
Var x, a, y:  real;   (описание переменных параметров)
Begin
write (‘ Введите x=’); (Ввод параметра x с клавиатуры)
readln (x);
```

```
a:=3.5*b/(b+exp(3*ln(c)));
if x < 1 then y:=a+b*x;       (расчет параметра y)
if x = 1 then y:=sqrt(abc(a+b/x)) else
y:=sqr(a*x+b*x);
writeln ('y=', y);       (вывод параметра y на экран)
End.
```



3. Входим в среду Pascal ABC, используя иконку на рабочем столе компьютера.

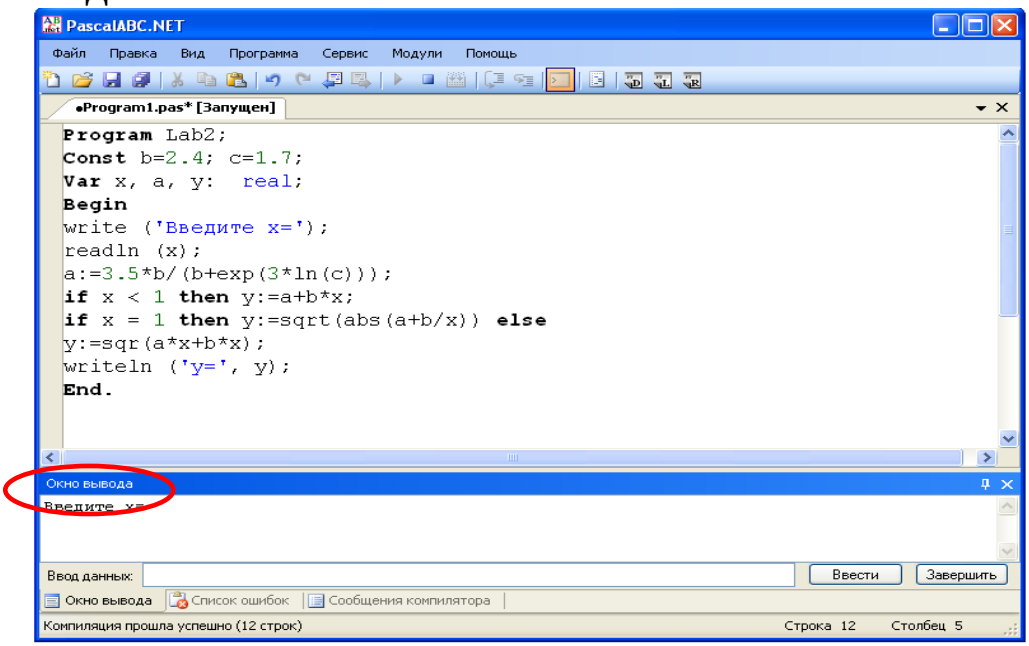
6. В верхнем окне браузера набирается текст программы на языке Паскаль:



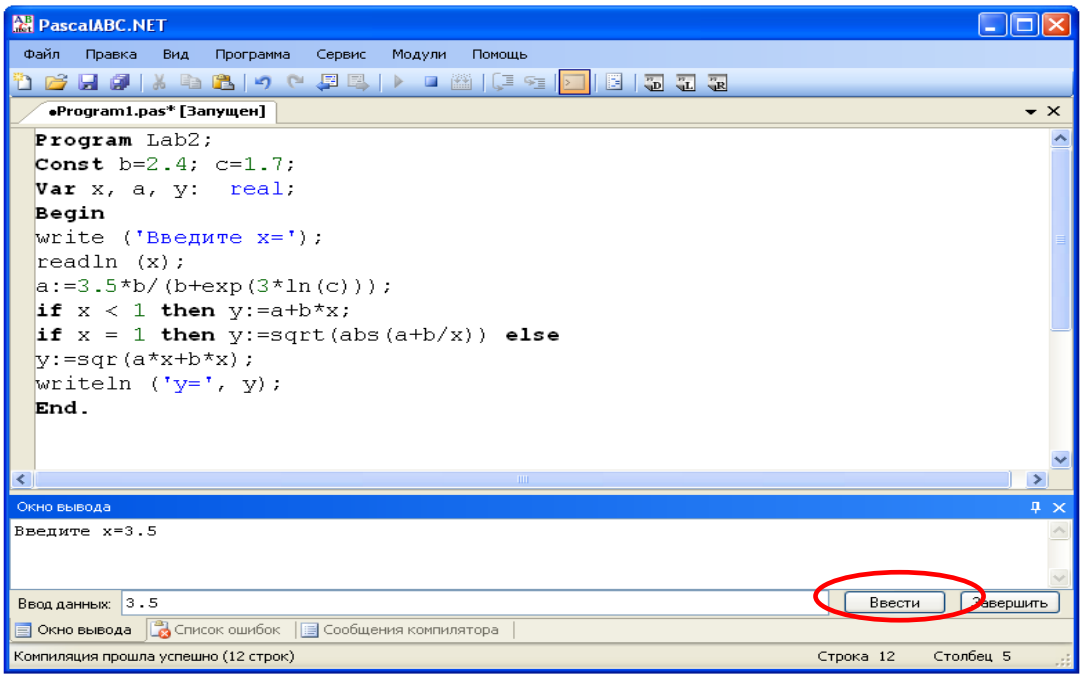
5. Сохраняем текст программы (подробно последовательность действий описана в предыдущем примере).

6. Воспользовавшись иконкой  , выполняем программу.

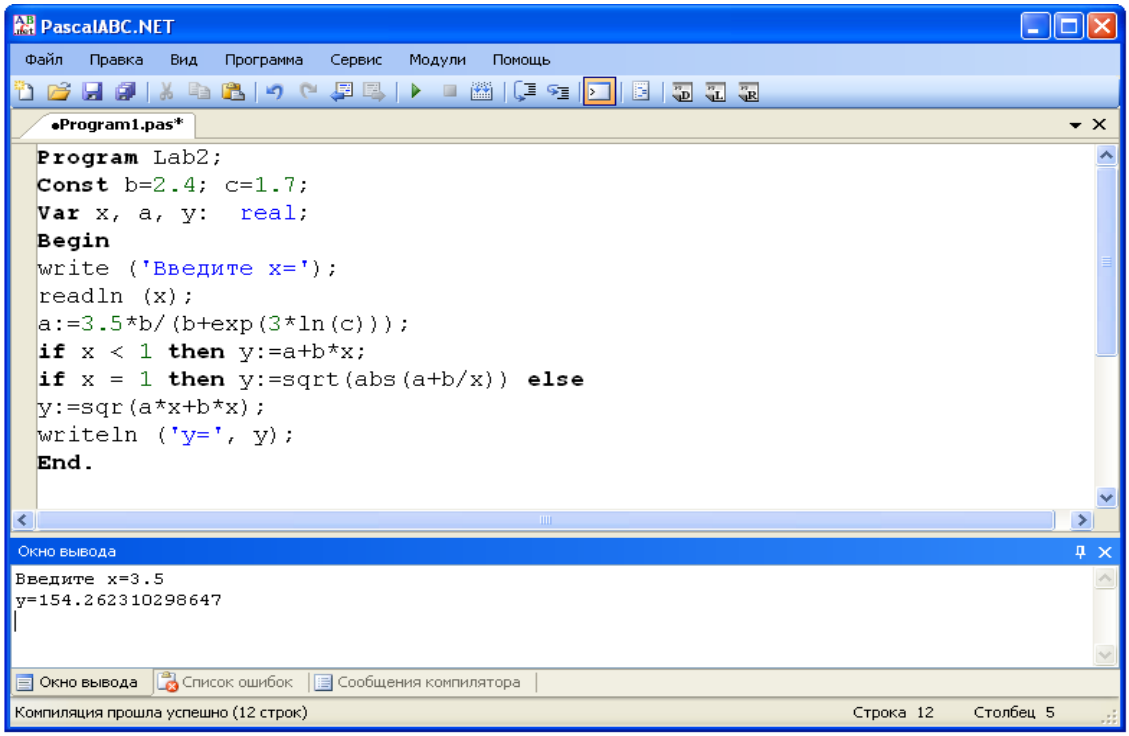
7. При возникновении ошибок, в Окне вывода будет появляться подсказка. Следует отлаживать программу (исправлять ошибки), пока в Окне вывода не появится дополнительная строка для ввода значения *x*:



Произвольно задайте значение *x* (например, 3,5) и нажмите «Ввести»:



После нажатия клавиши «Ввести» в окне вывода появится результат вычисления величины *y*:



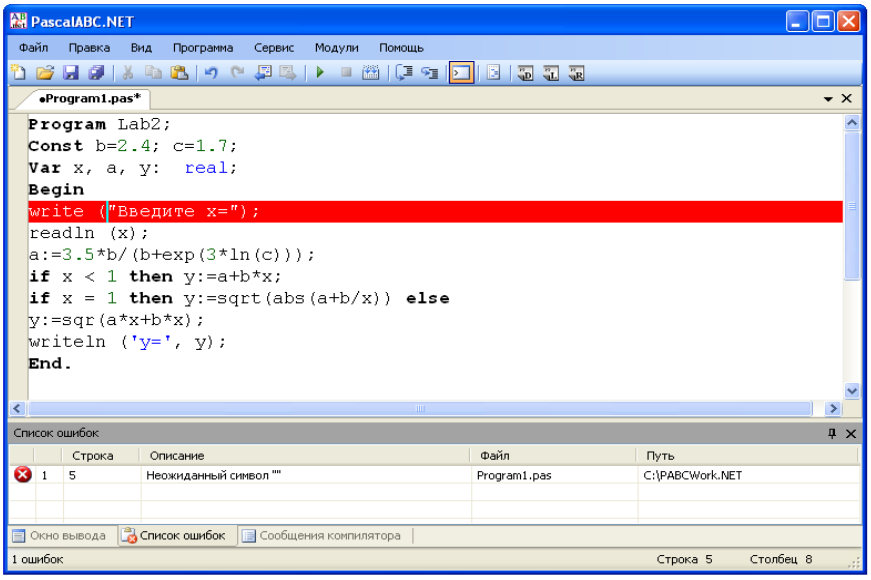
Повторите несколько раз вычисления, изменяя значение *x* в строке ввода.

Разъяснения по наиболее часто встречающимся ошибкам

Основные ошибки, встречающиеся в программах, описаны в предыдущем примере. В данном разделе приведены те ошибки,

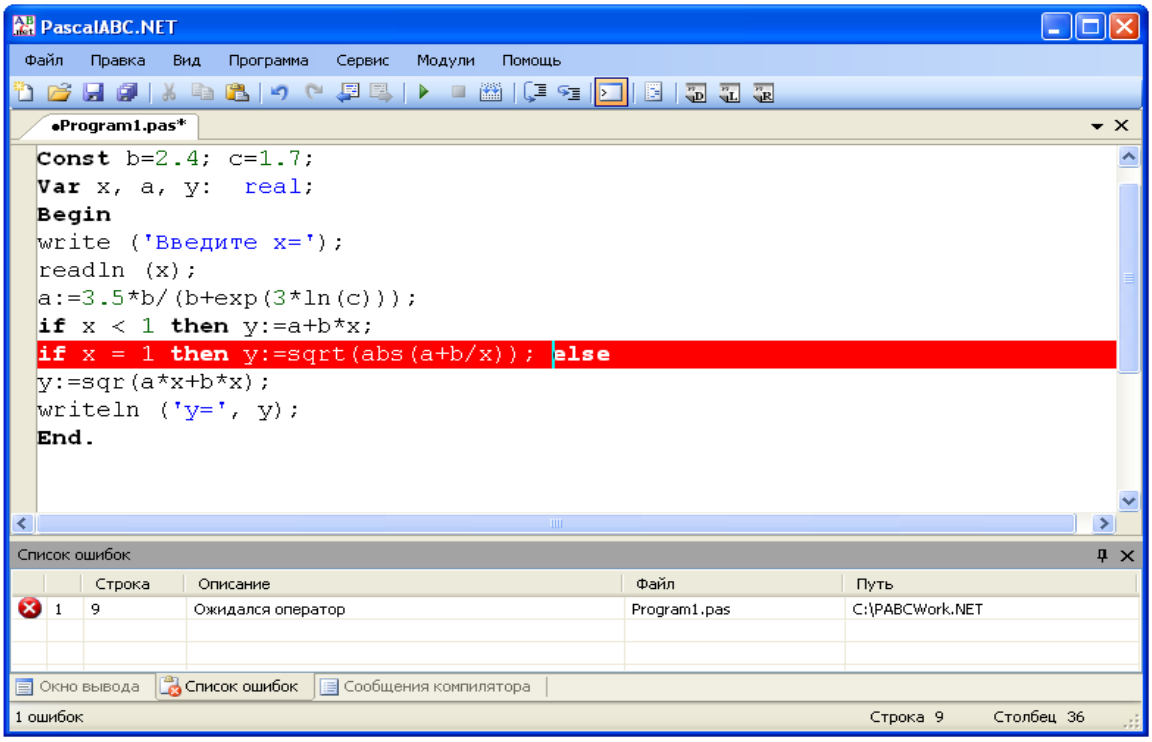
которые могут дополнительно возникнуть при выполнении текущей лабораторной работы.

1. Неправильное введение апострофов



Апострофы в программе необходимо вводить символом ‘. Двойной апостроф “ не допускается.

2. Неправильное введение условного оператора



Все составные части условного оператора написаны правильно, однако перед зарезервированным словом *else* стоит точка с запятой, что не допустимо. Точка с запятой ставится лишь после закрытия условного оператора или его началом.

Последовательность работы при незавершенной отладке приведена в предыдущем примере.

Задания для практического занятия

Задание № 2-П. Составить алгоритм, блок-схему и программу на языке Паскаль для вычисления значения функции $y = f(x)$ при заданном значении x, которое вводится с клавиатуры, на основе разветвляющихся алгоритмов с использованием условного оператора *if...then...else*. Запустив программу дважды, получить ответ для каждого из заданных значений x . Результат вывести с тремя знаками после запятой. Варианты заданий приведены в таблице

1. $y = \begin{cases} 2ax+1 & x > 2 \\ 3ax; & x < 2 \\ ax^2+3 & x = 2 \end{cases}$

2. $y = \begin{cases} 2{,}7x + 3\sqrt{x-1{,}2^x}, & x < 1 \\ 0{,}5x + \ln\left|x+1{,}2\right|, & 1 \le x = 3 \\ \sqrt[3]{x^2 + tg(\sqrt[3]{x^2}+1{,}2x)}, & x > 3 \end{cases}$

3. $y = \begin{cases} \log_3(x^2+4), & x < 2 \\ 4 - x^3, & x = 2 \\ \arcsin, & x < 2 \end{cases}$

4. $y = \begin{cases} 5/x + 3x^2 + 3x^3 + \cos x, & \left|a\right| > x \\ \sin\left|2a - x^2\right|, & \left|a\right| = x \\ \sqrt[5]{x^3 + tg^2\sqrt{x-5a}}, & \left|a\right| < x \end{cases}$

5. $y = \begin{cases} 3x^2 + 4\sqrt{a-x}, & a > x \\ \sin\sqrt{3a+x}, & a = x \\ e\sqrt{x-a}, & a < x \end{cases}$

6. $y = \begin{cases} \ln x & x > 0 \\ 0 & x = 0 \\ \ln(-x), & x < 0 \end{cases}$

7. $y = \begin{cases} \sin^2 x, & \left|x\right| < n/4 \\ \sin(tgx), & \left|x\right| \ge n/4 \end{cases}$

8. $y = \begin{cases} 3^{x-1} & x > 1 \\ (x-1)3, & x < 1 \\ 0, & x = 1 \end{cases}$

9. $y = \begin{cases} \log_3 x, & x > o \\ \sqrt{-x}, & x \le 0 \end{cases}$

10. $y = \begin{cases} \sqrt[3]{x^2-8}, & x < 0 \\ x+1 & x = o \\ \log_2 x, & x > o \end{cases}$

11. $y = \begin{cases} \ln\left|x^2+50\right|, & x < 0 \\ \frac{4}{5}(\sqrt[3]{x+\sqrt{x^2+1}}), & 0 \le x \le 15 \\ 3x^2 + \sin l^{x,} & x > 15 \end{cases}$

12. $y = \begin{cases} l^{\sin\left|x-1\right|}, & x = \frac{n}{2} \\ \cos\left|x - \frac{n}{2}\right|, & x > \frac{n}{2} \\ \sqrt{tg(\frac{3n}{2}x)}, & x < \frac{n}{2} \end{cases}$

13. $y = \begin{cases} \ln(x-1), & x > 1 \\ 0, & x < 1 \end{cases}$

14. $y=\begin{cases}\ln\left|x^2+25\right|, & x<10\\ \frac{4}{5}(\sqrt{x+\sqrt{x^2+1}}), & 10\le x\le 15\\ 3x^2+\sin l^{2x}, & x>15\end{cases}$

15. $y=\begin{cases}x^2-a^{2,}, & |x-a|<1\\ x^2+a^2, & |x-a|=1\\ x^2-2ax+1, & |x-a|>1\end{cases}$

16. $y=\begin{cases}\ln\left|\sin x+1\right|, & x<\frac{n}{6}\\ \cos\left|2x+3n\right|, & x=\frac{n}{6}\\ tgx^2+arctgx, & x>\frac{n}{6}\end{cases}$

17. $y=\begin{cases}ax^3+3\ln|a+x|, & |ax|<2\\ \sqrt{ax+\sqrt{a^2+x^2}}, & |ax|=2\\ e^{\sqrt{ax+\sin x}}, & |ax>2|.\end{cases}$

18. $y=\begin{cases}\sqrt{x-3x+4{,}3}, & x<1\\ 2\sin x+x^3+1, & x\ge 2\\ \sqrt{1{,}5x+x^2}, & 1\le x<2\end{cases}$

19. $y=\begin{cases}arctg\frac{x}{\sqrt{3x+x^2+1}}, & x<\frac{n}{2}\\ 2\sqrt{x+\sin(x-2)}, & x=\frac{n}{2}\\ tg\sqrt{x+1}, & x>\frac{n}{2}\end{cases}$

20. $y=\begin{cases}2^x+\sin|x-a|, & |a-x|<1,\\ e^{\sqrt{3a+x^2+3}}, & 1\le|a-x|\le 2,\\ 3x(x^2+3{,}5)+\log_3 x, & |a-x|>2\end{cases}$

21. $y=\begin{cases}\sqrt{x+\cos x+10^{-6,7}}, & x<4,\\ \sqrt{13a-2bx+x^2}, & 4\le x\le 10\\ a^2-e^2+|x|+\ln x, & x>10\end{cases}$

22. $y=\begin{cases}x^2+\sqrt[3]{x+10^{-1,6}}, & x<-3\\ \sin x+\ln|x|, & -3\le x\le -2\\ x^2+\sqrt{e^{-x^2}-x^2+2}, & x>-2\end{cases}$

23. $y=\begin{cases}2\sqrt{\sin x+tgx^2}, & 1\le x<3\\ x^3-\sqrt[6]{x+xa}, & x\ge 3\\ 1+\frac{x}{a}+abc, & x<1\end{cases}$

24. $Z=\begin{cases}\sqrt[3]{\sin(x-10^{-1,35}}, & x<0{,}3\\ \ln(1+\left|\frac{x}{y}\right|2{,}5), & |x|=0{,}3\\ e^{x+y}+|x+y|, & |x|>0{,}3\end{cases}$

25. $y=\begin{cases}x^b+\sin(x+a), & x>0,\\ x^5=x^2+2x+7, & x=0,\\ \sqrt{\sin\frac{n}{6}+\ln|3x|}, & x<0\end{cases}$

26. $y=\begin{cases}\cos\frac{x}{\sqrt{3x+x^2+1}}, & x<\frac{1}{2}\\ 2^{\sqrt{x+\sin(x-2)}}, & x=\frac{1}{2}\\ ctg\sqrt{x+1}, & x>\frac{1}{2}\end{cases}$

27. $y=\begin{cases}x^3+\sqrt[4]{\sin x+10^{1,6}}, & x<-3\\ \cos x+\sin|x|, & -3\le x\le -2\\ e^{-x^2}+\sqrt{e^{-x^2}+x^2+2}, & x>-2\end{cases}$

где $a=\frac{3{,}5-c}{bc}$; $b=7{,}2$; $c=8{,}7$; $n-$номервариантa.

Задания для самостоятельной работы

Задание № 2-С. Составить алгоритм, блок-схему и программу на языке Паскаль для вычисления значения функции $y = f(x)$ при заданном значении x, которое вводится с клавиатуры, на основе разветвляющихся алгоритмов с использованием условного оператора *if…then…else*. Запустив программу дважды, получить ответ для каждого из заданных значений x . Результат вывести с тремя знаками после запятой. Варианты заданий приведены в таблице.

Вариант	$y = f(x)$	Исходные данные	
		const	x
1	$y=\begin{cases} b+2\ln\lvert x\rvert & \text{при } x\le 3,\\ \dfrac{x^2}{x^2+a} & \text{при } x>3 \end{cases}$	$a=10{,}2$ $b=13{,}4$	1) $x=4{,}5$ 2) $x=1{,}72$
2	$y=\begin{cases} a+\dfrac{1}{2}e^{-x} & \text{при } x>0,\\ \cos(bx+1) & \text{при } x\le 0 \end{cases}$	$a=8{,}53$ $b=17{,}1$	1) $x=2{,}5$ 2) $x=-3{,}1$
3	$y=\begin{cases} \dfrac{1}{a^2+x^2} & \text{при } x\le 1,\\ b\cdot\ln\lvert x\rvert & \text{при } x>1 \end{cases}$	$a=7{,}2$ $b=5{,}7$	1) $x=2{,}92$ 2) $x=-3{,}57$
4	$y=\begin{cases} \dfrac{a+x^2}{b+\ln(\lvert x\rvert+1)} & \text{при } x\le 2,\\ e^x+x^2 & \text{при } x>2 \end{cases}$	$a=9{,}1$ $b=3{,}6$	1) $x=5{,}41$ 2) $x=-0{,}71$
5	$y=\begin{cases} a\sin^2 x+\sqrt{x} & \text{при } x\le 1,\\ be^{x^2} & \text{при } x>1 \end{cases}$	$a=1{,}1$ $b=3{,}2$	1) $x=4{,}23$ 2) $x=0{,}93$
6	$y=\begin{cases} a\cdot\operatorname{tg}(x^2) & \text{при } x\le -1,\\ b+\dfrac{x^2}{x^2+a} & \text{при } x>-1 \end{cases}$	$a=9{,}5$ $b=3{,}8$	1) $x=-4{,}52$ 2) $x=1{,}83$
7	$y=\begin{cases} (a+x)\operatorname{arctg}(ax) & \text{при } x>0,\\ \cos^2(b+x^3) & \text{при } x\le 0 \end{cases}$	$a=4{,}1$ $b=2{,}9$	1) $x=6{,}81$ 2) $x=-4{,}17$
8	$y=\begin{cases} \sin^3(a+x) & \text{при } x<5,\\ \ln\sqrt{\lvert b-x\rvert} & \text{при } x\ge 5 \end{cases}$	$a=1{,}9$ $b=3{,}4$	1) $x=7{,}39$ 2) $x=0{,}62$
9	$y=\begin{cases} \sqrt{1+x\sqrt{ax}} & \text{при } x\ge 2,\\ \sin(bx)+3 & \text{при } x<2 \end{cases}$	$a=4{,}6$ $b=3{,}2$	1) $x=3{,}78$ 2) $x=1{,}54$
10	$y=\begin{cases} \sqrt{e^{2x-b}}-1 & \text{при } x\le 0,\\ \dfrac{1}{x^2+a} & \text{при } x>0 \end{cases}$	$a=6{,}7$ $b=1{,}8$	1) $x=-0{,}24$ 2) $x=2{,}13$

5. Выполнение циклических операций. Массивы, операторы цикла, действия с матрицами

Цель занятия: Научиться составлять программы на языке Паскаль на основе циклических алгоритмов с использованием операторов цикла *while..do* и *repeat…until.* Научится работать с матрицами.

ЦИКЛ – ПОКА (С ПРЕДУСЛОВИЕМ)

Формат оператора:

WHILE <условие>
DO
Begin
< тело цикла>
End

Циклом называется последовательность действий, выполняемых многократно, каждый раз при новых значениях параметров.

Цикл с предусловием используется, когда неизвестно количество повторений

Выполняется следующим образом:

Сначала проверяется условие. Если оно истинно, то выполняется тело цикла. Если условие становится ложным, то тело цикла не выполняется, а выполняется следующий за END оператор. Таким образом, если условие с самого начала ложно, то тело цикла не выполнится ни разу.

Для того, чтобы избежать зацикливания программы необходимо обеспечить изменение на каждом шаге цикла значения хотя бы одной переменной, входящей в условие цикла. После выхода из цикла со сложным условием (с использованием операций **and**, **or**, **xor**) как правило, необходима проверка того, по какому условию цикл завершен

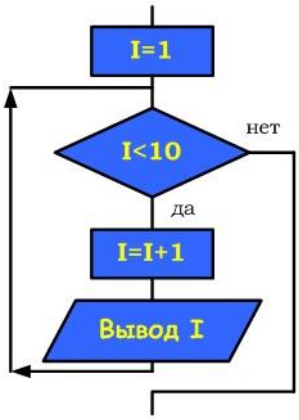
Тело цикла с предусловием

ПРИМЕР 2. Пары неотрицательных вещественных чисел вводятся с клавиатуры. Посчитать произведение для каждой пары и сумму всех чисел.

```
program cycle_while;
var x,y,sum:real;
otv:char;
begin
sum:=0;
otv='Д';
while (otv='Д') or (otv='д') do
begin
write('Введите числа x,y > 0 ');
```

выполняется пока условие истинно

ПРИМЕР 1.




```
I:=1;
WHILE I<10 DO BEGIN
   I:=I+1;
   WRITELN(I);
END;
```

```
readln(x,y);
writeln('Их произведение =
',x*y:8:3);
sum:=sum+x+y;
write('Завершить программу
(Д/Н)? ');
readln(otv);
end;
writeln('Общая сумма =
',sum:8:3);
readln
end.
```

ПРИМЕР 3. Нахождение наибольшего общего делителя двух целых чисел с помощью Алгоритма Эвклида.

```
program Evklid;
var a,b,c:integer;
begin
write('введите два целых числа : ');
readln(a,b);
while b<>0 do
begin
c:=a mod b;
a:=b;
b:=c;
end;
writeln('наибольший общий делитель = ',a);
readln
end.
```

ПРИМЕР 4. Подсчитать количество нечетных цифр в записи натурального числа *n*.

Идея решения. Из заданного числа выбирать из младшего разряда цифру за цифрой до тех пор, пока оно не исчерпается, т.е. станет равным нулю. Каждую нечётную цифру учитывать.

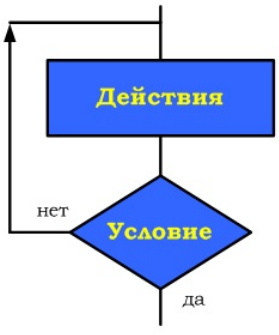
1. Ввести число n
2. K := 0 {подготавливаем счётчик}
3. Если n = 0, переход к шагу 7

4. Если n mod 10 mod 2 = 1, то K := K +1
5. n := n div 10
6. Переход к шагу 3
7. Вывод K
8. Конец

ПРИМЕР 5. Дана последовательность, общий член которой определяется формулой a_n=(n-1)/(n*n). Вычислить при *n*>2 сумму тех ее членов, которые больше заданного числа k. При решении задачи находится очередной член последовательно и, если он больше k, добавляется к сумме.
Решение:
1. Ввести k
2. S := 0
3. A := 1/4
4. n := 3
5. Сравнить A с k. Если A>= k, переход к шагу 10
6. S := S + A
7. A := (n-1)/(n*n)
8. n := n + 1
9. Переход к шагу 5
10. Вывод S
11. Конец

ЦИКЛ ДО (С ПОСТУСЛОВИЕМ)

Цикл предназначен для организации многократного исполнения набора инструкций (операторов, наименьшая автономная часть языка программирования). Если заранее неизвестно число повторений цикла, то можно использовать **цикл с постусловием**.



Формат оператора:

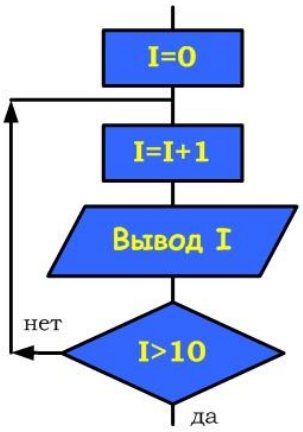
REPEAT
Оператор1;
Оператор2;
...
ОператорN;
UNTIL <условие>;

В большинстве процедурных языков программирования цикл с постусловием реализуется оператором **while**, отсюда его второе название– while-цикл.

Выполняется следующим об-Тело цикла с постусловием вы-
разом: Сначала выполняется те-**полняется пока условие <u>ложно</u>**
ло цикла, затем проверяется условие. Если оно ложно, то выполняется тело цикла. Если условие истинно, то цикл считается выполненным.

В этом цикле логическое выражение - **это условие выхода из цикла.** При описании циклов с постусловием необходимо принимать во внимание следующее:

ПРИМЕР 1




```
I:=0;
REPEAT
    I:=I+1;
    WRITELN(I);
UNTIL I>10;
```

- перед первым выполнением цикла условие его окончания (или продолжения) должно быть определено;
- тело цикла должно содержать хотя бы один оператор, влияющий на условие окончания (продолжения), иначе цикл будет бесконечным;
- условие окончания цикла должно быть в результате выполнено.

ПРИМЕР 2. Пары неотрицательных вещественных чисел вводятся с клавиатуры. Посчитать произведение для каждой пары и сумму всех чисел.

Решение:

```
program cycle_repeat;
var x,y,sum:real;
otv:char;
begin
sum:=0;
repeat
write('Введите числа x,y > 0 ');
readln(x,y);
writeln('Их произведение = ',x*y:8:3);
sum:=sum+x+y;
write('Завершить программу (Д/Н)? ');
readln(otv);
until (otv='Д') or (otv='д');
```

```
writeln('Общая сумма = ',sum:8:3);
readln
end.
```

ПРИМЕР 3. Подсчитать количество нечетных цифр в записи натурального числа *n*.

Идея решения. Из заданного числа выбирать из младшего разряда цифру за цифрой до тех пор, пока оно не исчерпается, т.е. станет равным нулю. Каждую нечётную цифру учитывать.

Решение:

1. Ввести число n
2. K := 0 {подготавливаем счётчик}
3. Если n mod 10 mod 2 = 1, то K := K +1
4. n := n div 10
5. Если n = 0, переход к шагу 7
6. Переход к шагу 3
7. Вывод K
8. Конец

ПРИМЕР 4. Составить программу планирования закупки товара в магазине на сумму, не превышающую заданную величину.

Решение. Обозначим через *x*, *k* – соответствующую цену и количество товара, через p – заданную предельную сумму, через s – общую стоимость покупки. Начальное значение общей стоимости покупки (*S*) равно нулю. Значение предельной суммы считывается с клавиатуры. Необходимо повторять запрос цены и количества выбранного товара, вычислять его стоимость, суммировать ее с общей стоимостью и выводить результат на экран до тех пор, пока она не превысит предельную сумму p. В этом случае на экран нужно вывести сообщение о превышении.

```
Program E_10;
Var x, k, p, s : Integer;
Begin
WriteLn('Введите цену товара и его количество');
ReadLn(x,k);
s:=s+x*k;
WriteLn('Стоимость покупки равна ',s);
Until s>p;
```

WriteLn('Суммарная стоимость покупки превысила предельную сумму');
End.

ЦИКЛ С ПАРАМЕТРОМ

Цикл с параметром используется, когда известно начальное значение переменной, конечное значение и шаг изменения равен 1 или –1, т.е. параметр увеличивается или уменьшается на единицу. Таким образом, цикл с параметром организует выполнение одного или нескольких операторов заранее определенное число раз (известное заранее)

Формат оператора:

Шаг изменения = 1 (по умолчанию)

```
FOR I:= N to K DO
Begin
< тело цикла>
End;
```

Шаг изменения = -1

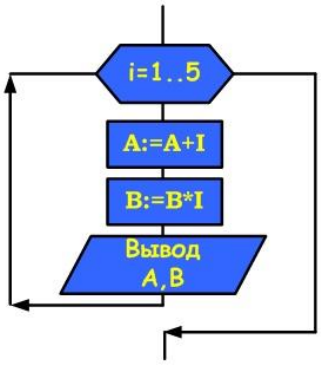
```
FOR I:= N to K DOWNTO
Begin
< тело цикла>
End;
```

I – параметр цикла
N – начальное значение параметра
K – конечное значение параметра

Выполняется следующим образом:

1. Параметру присваивается начальное значение N;
2. Проверка: Если значение параметра не больше (не меньше) конечного значения K, то переход на п.3 иначе п.6.
3. Выполняется тело цикла
4. Параметр цикла увеличивается (уменьшается) на 1

ПРИМЕР 1




```
FOR I:=1 TO 5 DO
   BEGIN
      A:=A+I;
      B:=B*I
      WRITELN(A,B);
   END;
```

ПРИМЕР 2. Найти произведение первых k натуральных чисел, кратных трём.

Решение:

5. Переход на п.2
6. Выход из цикла

Использовать цикл **for** необходимо при заранее известном количестве повторений. Нельзя изменять параметр в теле цикла. При использовании кратных (вложенных) циклов применять разные переменные в качестве параметров. Определять до цикла значения всех используемых в нем переменных. Не ставить точку с запятой после **do**.

При составлении алгоритма учтем, что первое натуральное число, кратное 3, есть тройка, а все последующие больше предыдущего на 3.
1. Ввод k
2. P := 1 {здесь накапливаем произведение}
3. T := 0 {здесь будут числа, кратные 3}
4. I := 1
5. Если I > k, переход к шагу 10
6. T := T + 3
7. P := P * T
8. I := I + 1
9. Перейти к шагу 5
10. Вывод P
11. Конец

ПРИМЕР 3. Вводятся 10 чисел, посчитать среди них количество положительных.

```
program cycle_for1;
var i,kn:byte; x:real;
begin
kn:=0;
for i:=1 to 10 do
begin
writeln('Введите ',i,' число: ');
readln(x);
if x>0 then kn:=kn+1 {увеличиваем количество на 1}
end;
writeln('Вы ввели ',kn,' положительных чисел.');
readln
end.
```

ПРИМЕР 4. Напечатать буквы от 'Z' до 'A'.

```
program cycle_for2;
var c:char;
begin
```

```
for c:='Z' downto 'A' do write(c);
readln
end.
```

ПРИМЕР 5. Вычислить N-е число Фиббоначчи. Числа Фиббоначчи строятся следующим образом: F(0)=F(1)=1; F(i+1)=F(i)+F(i-1); для i>=1. Это пример вычислений по рекуррентным формулам.

```
program Fib;
var a,b,c:word; i,n:byte;
begin
write('введите номер числа Фиббоначчи ');
readln(N);
a:=1; {a=F(0), a соответствует F(i-2)}
b:=1; {b=F(1), b соответствует F(i-1)}
for i:=2 to N do
begin
c:=a+b; {c соответствует F(i)}
a:=b; b:=c; {в качестве a и b берется следующая пара чисел}
end;
writeln(N,'-е число Фиббоначчи =',b); {для N>=2 b=c}
readln
end.
```

Пример №3. Вычислить значение:

$$y = \sum_{k=0}^{8} 5a_k b + 4{,}3\prod_{j=1}^{4}\sqrt{c_j}$$

b=0,5; a_0=12; a_k=20; h=2; *c(4)*=6,7,8,9.

Для организации цикла при вычислении суммы использовать оператор цикла *while..do*

Последовательность работы

1. Для упрощения расчетов разобьем выражение на две составляющие:

$$k = \sum_{k=0}^{8} 5a_k b, \qquad f = \prod_{j=1}^{4}\sqrt{c_j},$$

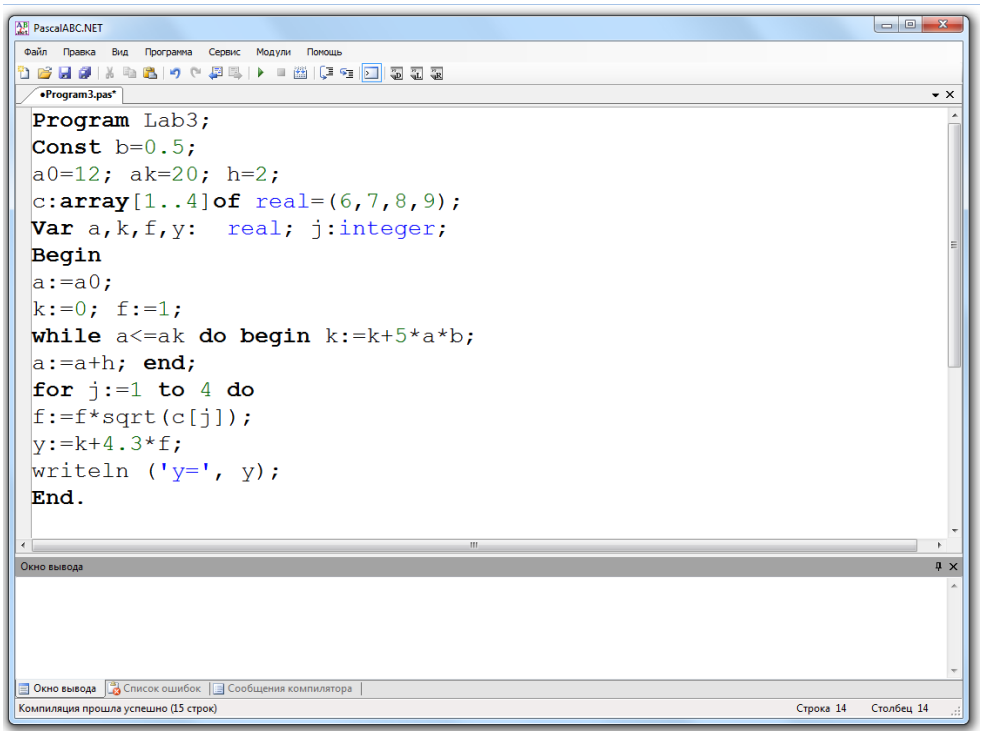
2. Составляем программу расчета, используя операторы цикла *while..do u for.*

Текст программы:

```
Program Lab3;
Const b=0.5;       {описание постоянных параметров}
a0=12; ak=20; h=2;
c:array[1..4]of real=(6,7,8,9); {описание массива}
Var a,k,f,y:  real; j:integer; {описание переменных}
Begin
a:=a0;  k:=0; f:=1;
while a<=ak do begin k:=k+5*a*b;   {расчет суммы}
a:=a+h; end;
for j:=1 to 4 do          {обращение к массиву}
f:=f*sqrt(c[j]);          {расчет произведения}
y:=k+4.3*f;               {расчет параметра y}
writeln (‘y=’, y);     {вывод параметра y на экран}
End.
```

3. Входим в среду Pascal ABC, используя иконку на рабочем столе компьютера.

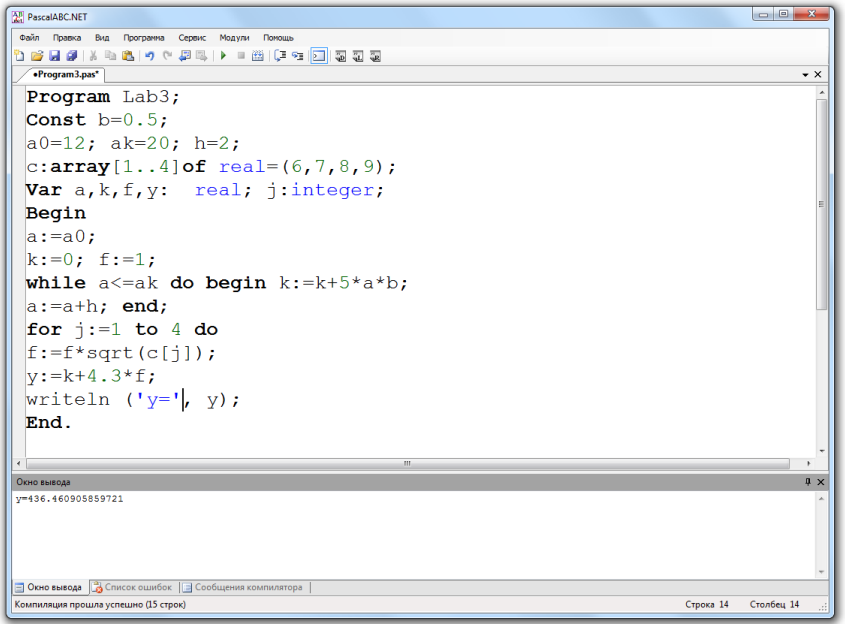
4. В верхнем окне браузера набирается текст программы на языке Паскаль:



5. Сохраняем текст программы (подробно последовательность действий описана в примере №1).

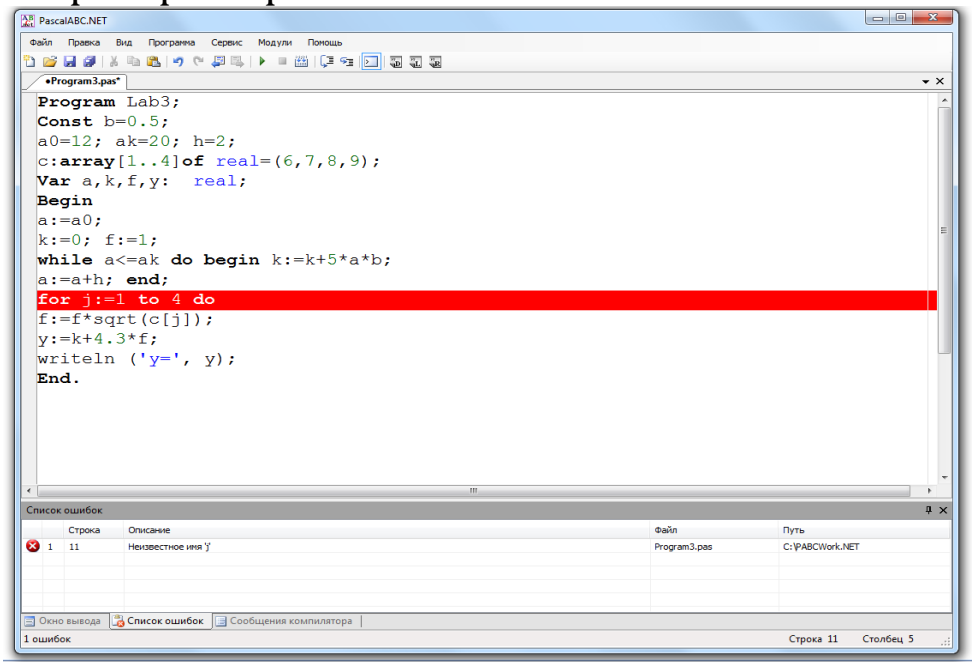
6. Воспользовавшись иконкой , выполняем программу.

7. При возникновении ошибок, в Окне вывода будет появляться подсказка. Следует отлаживать программу (исправлять ошибки), пока в Окне вывода не появится результат:

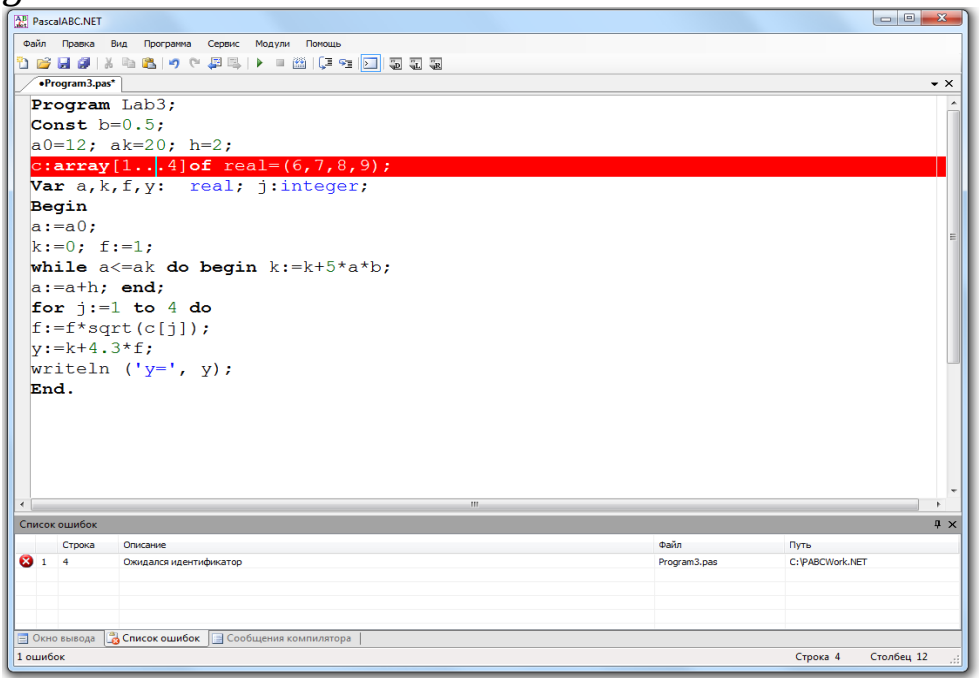


Разъяснения по наиболее часто встречающимся ошибкам

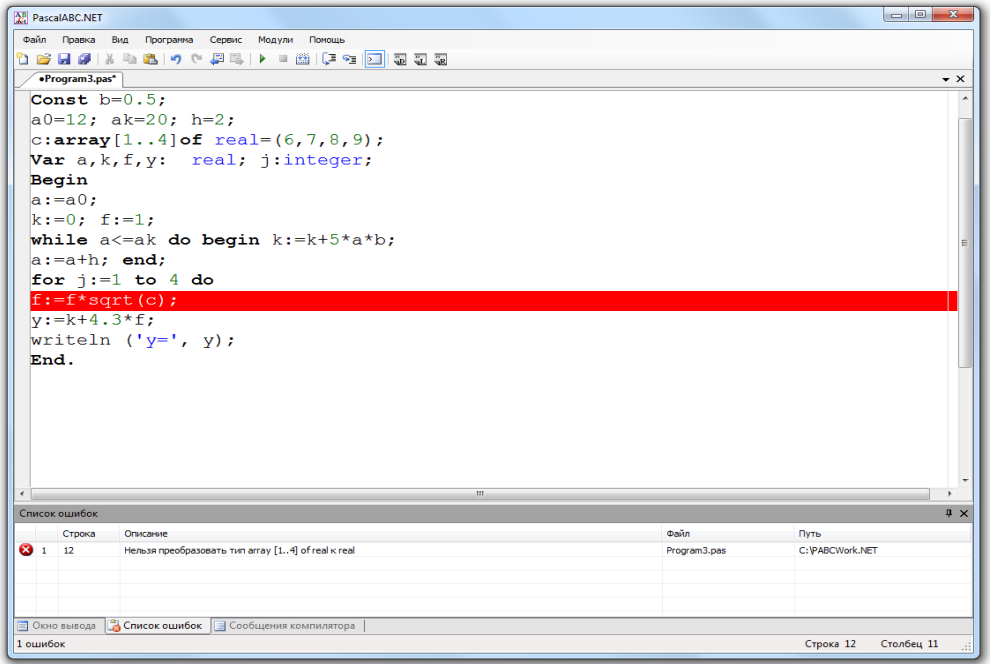
Основные ошибки, встречающиеся в программах, описаны в предыдущих примерах. В данном разделе приведены те ошибки, которые могут дополнительно возникнуть при выполнении текущей лабораторной работы.



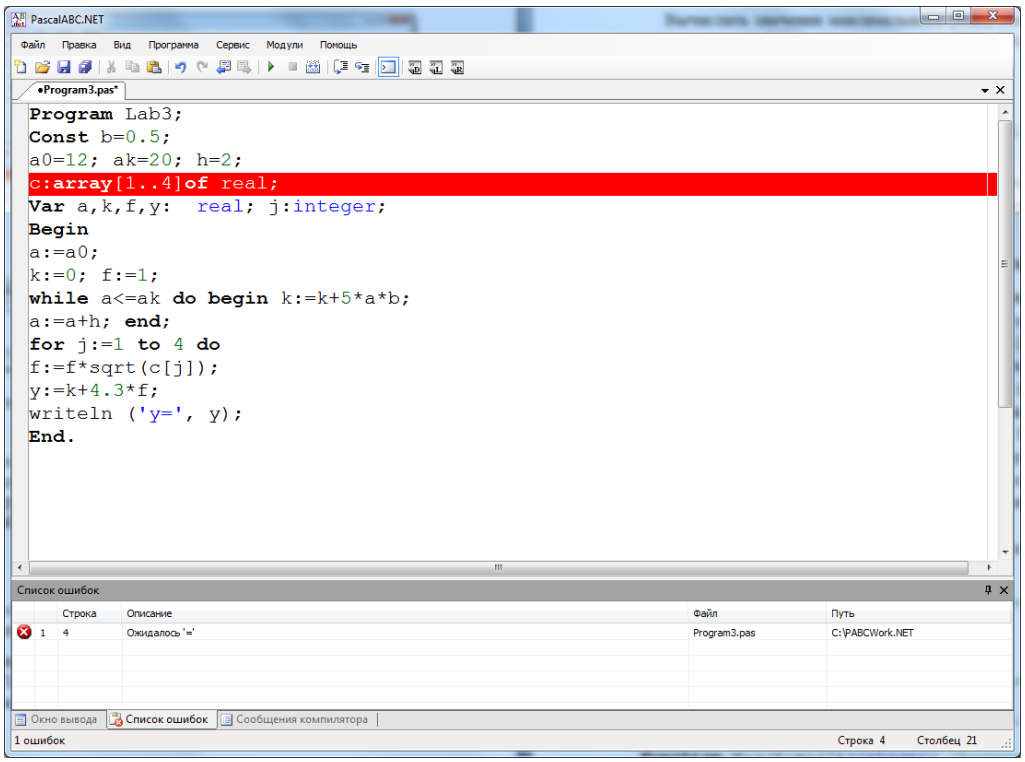
Номер элемента матрицы, обозначенный за *j*, не описан в программе. Следует описать его в разделе ***Var***, следующим образом *j:integer*.



При описании размерности массива в квадратных скобках следует ставить две точки (..), вместо указанных в программе трех (...).



В программе между параметрами должно быть соответствие по типу данных. Так как в разделе описаний данный параметр (С) описан как массив, то и при вычислении необходимо указать, что это параметр содержит несколько значений С[j].



При описании массива в разделе описания ***Const*** после указания типа необходимо поставив знак равно перечислить численные значения, входящие в массив, в круглых скобках:

c:array[1..4]of real=(6,7,8,9);

8. Последовательность работы при незавершенной отладке приведена в примере №1. Следует отлаживать программу (исправлять ошибки), пока в Окне вывода не появится результат

Задания для практического занятия

Задание № 3-П. Составить алгоритм, блок-схему и программу на языке Паскаль для вычисления функции *y* на основе циклических алгоритмов, используя операторы цикла *while..do* , *for* и *repeat.* Результат вывести с четырьмя знаками после запятой.

1. $y=\sum_{n=1}^{15}\frac{1}{(n+1)};$

2. $y=\sum_{i=1}^{N}\frac{\sin k}{(i+1)};$

3. $y=1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{(-1)^{n+1}}{n};$

4. $y=\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\ldots+\frac{1}{(n-1)\cdot n},n>2;$

5. $y=(1+\frac{1}{1^2})(1+\frac{1}{2^2})\cdot\ldots\cdot(1+\frac{1}{n^2}),n=5;$

6. $y=\sum_{k=1}^{n}\frac{1}{(2k-1)(2k+1)};$

7. $y=\sum_{k=1}^{n}\frac{2k+1}{(k+1)(k+2)};$

8. $y=\sum_{k=1}^{n}\frac{2k+1}{(2k^2+1)\cdot k};$

9. $y=\sum_{k=1}^{n}\frac{k+1}{k\sqrt{k^3+2}};$

10. $y=\sum_{k=1}^{n}\frac{8k}{3k^2+5};$

11. $y=\sum_{k=1}^{n}\frac{k^3}{2^k+3};$

12. $y=\sum_{k=1}^{n}\frac{3(k+1)}{7k^3+9};$

13. $y=\sum_{k=1}^{n}\frac{k+1}{\sin k+e^{k^2+1}+1};$

14. $y=\sum_{k=1}^{n}\frac{k^{k+1}}{a^{k+1}+(k+1)^a};$

15. $y=\sum_{k=1}^{n}\frac{(100-k)}{\lg k+5^{k^3}};$

16. $y=\sum_{m=1}^{n}\frac{(-1)^m\sqrt{m}}{c^{2m}};$

17. $y=\sum_{k=1}^{n}\frac{k^k+a^5}{\sqrt{3^k+k^3}};$

18. $y=\sum_{k=1}^{10}\frac{arc\quad\cos(\frac{k}{w})}{k^k+tg(k+1)};$

19. $y=\prod_{g=1}^{7}(-1)^g\frac{\cos(g^2+5)}{g^4+|g-71|};$

20. $y=\prod_{n=1}^{5}(-1)^{n+1}\frac{arc\quad tgn}{n^{1,6}-\ln(n+13)};$

21. $y=\prod_{n=1}^{6}\frac{n+3}{n^2+4n+1};$

22. $y=\prod_{i=1}^{7}\frac{i^4+i^2+3}{\sqrt{i^i+e^i}};$

23. $y=\prod_{n=1}^{5}\frac{3n+1}{n^2+5n+7};$

24. $y=\prod_{n=1}^{9}\frac{n^3+3n+1}{\sqrt[3]{n^2+7n+91}};$

25. $P=\prod_{k=1}^{4}\prod_{n=2}^{5}\frac{k(n+1)}{n(k+1)};$

26. $S=\sum_{k=1}^{3}\sum_{n=4}^{9}\frac{\sqrt{n^k+1}}{\log k^n};$

27. $y=\prod_{i=1}^{g}\sum_{m=1}^{2}(-1)^m\frac{(i^3+m^4)}{\sqrt{\ln(i+m)}+i^{\frac{i}{m}}};$

28. $y=\prod_{i=1}^{4}\sum_{m=1}^{5}(\frac{2\cdot i^m+4\cdot i\cdot m+tgi}{\sin m}-e^{-im}$

29. $y=\sum_{k=1}^{7}\sum_{m=1}^{4}\frac{\sqrt[4]{6k^m+m+7k}}{\ln(m+k)+m\cdot k}$

30. $y=\sum_{h=1}^{7}\prod_{m=1}^{5}\sqrt{\frac{k+m^3+e^m}{\log_m{}^k+(mk)^3}};$

31. $y=\prod_{k=1}^{8}\prod_{i=1}^{4}(-1)^i\frac{\sqrt{5i^4+e^k}}{\cos(i+k)^3+k^i};$

32. $y=\prod_{i=1}^{4}\sum_{m=1}^{21}(e^{\sqrt{i^2+m^{i+1}}}+\frac{i^3}{m^4+i^m})$

Задания для самостоятельной работы

Задание № 3-С. Составить алгоритм, блок-схему и программу на языке Паскаль, выполняющую табулирование функции $y = f(x)$ на отрезке $[x_1, x_n]$ с шагом h, на основе циклических алгоритмов, используя операторы цикла *while..do* , *for* и *repeat.* Результат вывести с четырьмя знаками после запятой. Варианты заданий приведены в таблице.

Вариант	$y = f(x)$	Исходные данные
1	$y = \frac{\sqrt{ax}}{b + ax\sqrt{x}}$	$x_1 = 1;\ x_n = 3;\ h = 0{,}2;$ $a = 3{,}5;\ b = 1{,}2$
2	$y = \sin(ax) + 3\cos^2(bx^2 + 1)$	$x_1 = 0;\ x_n = 5;\ h = 0{,}1;$ $a = 0{,}5;\ b = 0{,}7$
3	$y = \frac{1 + a(x + b)}{3 + \cos(ax)}$	$x_1 = 1;\ x_n = 3;\ h = 0{,}2;$ $a = 3{,}9;\ b = 2{,}3$
4	$y = bx\sqrt{1 + a^2 \ln x}$	$x_1 = 2;\ x_n = 3;\ h = 0{,}1;$ $a = 4;\ b = 7$
5	$y = \frac{b\cos x}{1 + a^2 \sin^3 x}$	$x_1 = 1;\ x_n = 6;\ h = 0{,}5;$ $a = 0{,}57;\ b = 9$
6	$y = a\left(\frac{b}{x} - \frac{\ln ax}{b^2}\right)$	$x_1 = 2;\ x_n = 5;\ h = 0{,}5;$ $a = 1{,}5;\ b = 4{,}8$
7	$y = \sqrt{1 + ax + b\cos x}$	$x_1 = 2;\ x_n = 8;\ h = 0{,}2;$ $a = 4{,}2;\ b = 1{,}5$
8	$y = ax(1 + a\,e^{-x})$	$x_1 = 2;\ x_n = 7;\ h = 0{,}5;$ $a = 3{,}5$
9	$y = b\ln(ax^2) + b\ln^2 x$	$x_1 = 1;\ x_n = 4;\ h = 0{,}3;$ $a = 4{,}3;\ b = 5{,}4$
10	$y = \frac{\ln(ax^2 + b)}{ax + 1}$	$x_1 = 2;\ x_n = 6;\ h = 0{,}4;$ $a = 1{,}4;\ b = 2{,}5$
11	$y = \frac{\cos(ax^2)}{1 + \mathrm{tg}^3(bx)}$	$x_1 = 0;\ x_n = 1;\ h = 0{,}1;$ $a = 2{,}1;\ b = 0{,}3$
12	$y = a\ln\frac{x}{bx^2 + 2}$	$x_1 = 3;\ x_n = 6;\ h = 0{,}3;$ $a = 1{,}9;\ b = 1{,}1$

6. Использование стандартных алгоритмов. Составление программ с использованием подпрограмм процедур и функций

Цель занятия: Получить представление об использовании подпрограмм на языке Паскаль, научиться программировать типовые алгоритмы, используемые при вычислении математических выражений.

Практически во всех алгоритмических языках имеется возможность программирования функций и процедур - блоков операторов, оформленных в виде подпрограмм. Разработка функций и процедур необходима при многократном использовании в разных местах программы или в нескольких программах блока операторов, выполняющих однотипные действия, например, расчет значений сложной функции при различных значениях аргумента.

Процедуры (подпрограммы) и функции, определяемые программистом, приводятся в разделе описания основной программы. Процедуры и функции имеют заголовок, раздел описания и раздел операторов.

Заголовок процедуры состоит из служебного слова **Procedure**, имени процедуры и списка параметров, например:

```
Procedure Name_P(p1, p2,...: "тип";  Var p3, p4,...: "тип";...);
```

Заголовок функции состоит из служебного слова **Function**, имени функции и списка параметров, кроме того указывается тип возвращаемого функцией значения, например:

```
Function Name_F("список формальных параметров"):"тип результата";
```

Здесь **Function и Procedure** - служебные слова,

Name_F, Name_P - имена функции и процедуры соответственно,

p1, p2 - имена формальных параметров-значений,

p3, p4 - имена формальных параметров-переменных,

. . . - многоточие означает возможность перечисления большего числа параметров.

В дальнейшем, если не оговаривается особо, все сказанное к процедуре относится также и к функции.

Тип возвращаемого функцией значения может быть простым, строковым или типом-указателем. **Тип формальных параметров** может быть любым, но должен указываться только идентификатором (именем типа). Таким образом, имя типа формального параметра - массива должно быть задано предварительно в операторе Type, например: **Type M= array[1..100]of real;** Затем тип массива может указываться в заголовке процедуры, например: **Procedure Name_P(p: M);** Тип формальных параметров описывается только в заголовке процедуры. Список формальных параметров может отсутствовать, например, процедура Randomize; не имеет параметров.

Если в результате выполнения нескольких операторов получается одно значение переменной, то эти операторы можно включить в описание функции. Например, функция sin(x); возвращает значение, которое присваивается переменной Y:=sin(x).

Если в результате выполнения нескольких операторов производится некоторое действие или расчет нескольких переменных, то эти операторы лучше включить в описание процедуры. Например, процедура ClrScr; из модуля CRT очищает экран.

Вызов процедуры осуществляется в разделе выполнения основной программы или других процедур (вложенные процедуры). Программа (процедура) внутри которой вызывается другая процедура называется внешней по отношению к вызываемой процедуре.

При вызове процедуры вместо формальных параметров подставляются фактические параметры, значения которых используются в процедуре. Например:

Name_P(p11, p22,..., p33, p44,...); - вызов процедуры **Name_P,**

Y:= Name_F("список фактических параметров"): - вызов функции **Name_F**,

Здесь **p11, p22, . . .** - имена или значения переменных,

p33, p44, . . . - имена переменных, значения которых возвращаются в программу.

Y - переменная, которой присваивается значение возвращаемое функцией.

Типы соответствующих формальных и фактических параметров должны совпадать, а имена могут совпадать или быть различными. Вместо параметров-значений можно подставлять имена переменных, значения переменных или выражения, вместо параметров-переменных подставляются имена переменных. Функция и параметры-переменные возвращают во внешнюю программу значения, полученные после окончания работы функции или процедуры. Изменения параметров-значений в процедуре носит локальный характер, во внешней программе соответствующие фактические параметры не изменяются. Если не требуется передавать во внешнюю программу новые значения, то следует использовать параметры-значения, а не параметры-переменные.

В процедуре можно использовать локальные метки, константы и переменные, описав их в разделе описания процедуры. Локальные имена не должны совпадать с именами формальных параметров, а их значения не передаются во внешнюю программу. Метки, константы и переменные, описанные во внешней программе раньше, чем сама процедура, называются **глобальными** по отношению к вызываемой процедуре. Если локальные и глобальные имена совпадают, то в процедуре используются локальные значения, а во внешней программе - глобальные значения, т. е. локальные и глобальные идентификаторы независимы. Если имя глобальной переменной уникально (в процедуре не описывается переменная с таким же именем) и ее значение в процедуре изменяется, то оно изменяется и во внешней программе. Вызывая в программе процедуру, программист использует ее имя и параметры не анализируя, а часто и не зная содержимого процедуры. Поэтому в целях универсальности процедур следует все значения в процедуру передавать через список параметров, а переменные внутри процедуры описывать, т. е. делать их локальными.

Приведем пример процедуры вывода на экран визитной карточки программиста.

```
Program NP_1;
Var Dat, Fam: string;  { Fam: глобальная переменная }
Procedure VIZ(D_R :string);  { D_R  - формальный параметр }
Var S_t: string;          { S_t: локальная переменная }
```

```
Begin   Writeln('| ---------------------------------- |');
      Writeln('| Разработчик программы:',   Fam:14,' |');
      Writeln('|                                    |');
      Writeln('| г. Самарканд          ',   D_R:14,' |');
      Writeln('| Телефон:  (+99894) 182-47-43              |');
      Writeln('| ---------------------------------- |');
      Write('   Комментарий: '); Readln(S_t)
end;
Begin   Fam:='А.Абдирашидов';  Dat:='06.12.95';{Dat - фактический
параметр }
     VIZ(Dat);  { вызов процедуры }         Readln   END.
```

Если процедура описана в другом файле с именем, например, F_PR. pas, то ее можно подключить к программе, указав в разделе описания директиву: **{$I F_PR. pas}**

В практических задачах часто пишутся процедуры, возвращающие значения элементов массивов. Приведем пример процедуры расчета "N" значений функции, например, Y= 4*sin(x)+7*cos(x); в заданном диапазоне x1<=x<=x2, при N<=100 и равномерной разбивке диапазона.

```
type r_1000= array[1. . 1000] of real;      { задается тип  r_1000 }
var Z: r_1000;  x1, x2: real;   n: word;
Procedure Mas_Y(var Y:r_1000; x1,x2:real; n:word); {Y - параметр-
                                        переменная}
var i: word;   x,  dx: real;          { локальные параметры }
begin
   If (n>1000) or (n<2) then begin
        writeln('Длина массива >1 и не должна превышать 1000');
        Readln;  Halt     end;
     i:=0;  x:=x1;  dx:=(x2-x1)/(n-1);  { dx - шаг изменения аргумента
}
   If dx<= 0 then begin
   writeln('x2 должно быть больше x1'); Readln;  Halt  end;
  While x<x2 do  begin
     i:= i+1;  x:= x1 + dx*(i-1);   Y[i]:= 4*sin(x)+7*cos(x)
  end
end;
```

```
begin Writeln('Введите значения x1,x2, (x2>x1)');  Readln(x1, x2);
    Writeln('Введите значение  1 <n<= 1000 ');   Readln(n);
  Mas_Y(Z, x1, x2, n);{вызов процедуры, возвращающей массив "Z"}
end.
```

Здесь тип формального параметра "Y" задается в разделе описания типов внешней программы и совпадает с типом фактического параметра "Z", значения элементов которого возвращаются во внешнюю программу.

Оператор **Halt** прерывает выполнение всей программы, даже если он используется внутри процедуры. Применение оператора **Exit** внутри процедуры вызывает прерывание процедуры, но не внешней программы.

Приведем пример процедуры вывода массива чисел в файл:

```
 Type  M_30x30_r=array[1..30,  1..30]  of  real;  {задается  тип
M_30x30_r}
var  x: M_30x30_r;    i, j, n, m: byte;
Procedure Wr_M(a: M_30x30_r;  name_f: string;  n, m: byte);
Var i, j: byte;              { a - массив NxM,   n<=30,  m<=30 }
     f: text;                      { name_f - имя файла }
begin   assign(f, name_f);  rewrite(f);
   For i:= 1 to n do begin writeln(f);
      For j:= 1 to m do write(f, a[i,j]:6:2)   end;
   close(f)
end;
Begin    N:= 10;                { создание симметричной матрицы }
  for i:= 1 to N do  for j:= i to N do x[i, j]:= 0.5 + random(50);  {запол-
нение верхней треугольной матрицы}
 for i:= 1 to N do  for j:= i to N do
    x[j,i]:= x[i,j];{заполнение нижней, симметричной части матрицы}
 Wr_M(x,'file_1.out',N,N);{вызов процедуры записи массива в файл}
end.
```

Для правильного считывания данных, записанных в файл бесформатным выводом необходима запись пробела для разделения чисел.

Написать и отладить программы с использованием процедур:

1. Вывести на экран визитную карточку программиста с указанием текущей даты.

2. Вывести на экран визитную карточку программиста с указанием времени.

Примечание к п. п. 1, 2 : рассмотреть случаи задания фамилии разработчика в качестве глобального, локального и формального параметра.

3. Рассчитать массив из N значений функции Y= ex + Cos(x) при изменении аргумента с постоянным шагом в диапазоне x1. . x2, и записи массива в файл. Значения N, x1, x2 и имя файла задаются оператором ввода

4. Рассчитать массив из N значений функции $Y=\ln(x) - x^3$ при изменении аргумента с постоянным шагом в диапазоне x1. . x2, и записи массива в файл. Значения N, x1, x2 и имя файла задаются оператором ввода.

5. Создать единичную матрицу NxN (N<=50). Элементы матрицы на главной диагонали равны единице, остальные - нулю. Вывести на экран массив 20x20.

6. Заменить элементы матрицы NxN (N<=30), расположенных в строках на элементы, расположенные в столбцах, т. е. **a[i, j]** на **a[j, i]**. Вывести на экран исходный и транспонированный массивы размером 10x10.

Приведем пример функции для расчета высоты треугольника по заданным значениям его сторон.

```
Program TR;
Var a, b, c, ha, hb, hc: real;
Function  H_TR(a, b, c: real): real;   {a, b, c - Стороны треугольника}
 Var p, s: real;
Begin
 If (a<0) or (b<0) or (c<0) Then  begin
    Writeln('Стороны треугольника >0 ?');  readln;  Halt  end;
 If (a>(b+c)) or (b>(a+c)) or (c>(a+b)) Then  begin
          Writeln('a<(b+c),  b<(a+c),  c<(a+b) ?');  readln;  Halt  end;
  p:= (a+b+c)/2;                                  { полупериметр }
  s:= Sqrt(p*(p-a)*(p-b)*(p-c));                          { площадь }
                    H_TR:= 2*s/a; { Присвоение функции значения }
```

```
End;
Begin
  Writeln('Введите значения сторон треугольника a,b,c');
Readln(a,b,c);
   ha:= H_TR(a, b, c);   hb:= H_TR(b, a, c);   hc:= H_TR(c, b, a);
  Writeln('Высоты  треугольника:');
  Writeln('ha=',ha:-10:4, 'hb=',hb:-10:4, 'hc=',hc:-10:4);  Readln
End.
```

В программе трижды вызывается функция расчета высоты треугольника для различных комбинаций фактических параметров, что позволяет вычислить все высоты треугольника.

Написать и отладить программы с использованием функций:

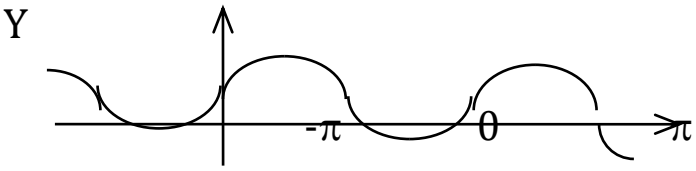
1. Рассчитать площадь треугольника по известным координатам вершин с использованием формулы **S= 0.5*abs(y1*(x3-x2)+ y2*(x1-x3) + y3*(x2-x1))**, а также разработать функцию расчета площади треугольника по формуле Герона (значения сторон вычисляются внутри функции). Вывести на экран значения площади треугольника, подсчитанные по обеим формулам (функциям).

2. Вывести на экран сообщения о типе треугольника: остроугольный, прямоугольный, тупоугольный (задаются координаты вершин). Для расчета углов использовать теорему косинусов, например: **c_a:= (b*b + c*c - a*a)/(2*b*c);**
где c_a - косинус угла, противоположного стороне "a".

3. Определить нахождение точки "A" внутри прямоугольника, включающего область из "N" точек с координатами Xi, Yi, i=1, . . . , N< 21. Стороны прямоугольника параллельны осям координат. Координаты точек задаются в основной программе функцией Random(1000). Функция возвращает значение логического типа.

4. Определяющую симметричность матрицы (NxN). Функция возвращает значение логического типа.

Приведем пример использования функции для расчета суммы членов степенного ряда, представляющего тригонометрическую функцию Y= Sin(x).




```
PROGRAM fun_sin;
var  y,  y1,  x1: real;
Function Sin_r( x: real): real;
2π          3π    X
Var a,  k,  y: real;  i: longint;
begin
  {if abs(x)>2*Pi Then x:= 2*pi*Frac(x/(2*Pi)); {учет периодичности}
   if abs(x) > 2*Pi Then x:= x - 2*pi*Int(x/(2*Pi));       { функции }
   if abs(x) > Pi Then x:= Pi - ABS(x);    {учет асимметрии функции}
        i:= 0;  a:= x;  y:= a;
    while abs(a)>0.0000001 do begin  i:=i+1; k:=-x*x/(2*i*(2*i+1));
                            a:= a*k;    y:= y + a    end;
       Sin_r:= y;                { присвоение функции ее значения }
end;
Begin
     write('Введите значение аргумента: x1= ');  Readln(x1);
  Y:= Sin_r(x1);     { вызов функции,  разработанной программистом
}
  Y1:= Sin(x1);               { вызов стандартной функции }
    writeln('значение аргумента: x1= ',  x1);
    writeln('расчетное значение функции: Sin_r(x1)= ',  y :-11:8);
     writeln('контрольный результат:     Sin(x1) =  ',  y1:-11:8);
     writeln('Нажмите Enter');    readln;   end.
```

В описании функции обязателен оператор присвоения функции ее значения. Изменение величины параметра-значения "x" в теле функции не отражается на его значении во внешней программе. Функция возвращает значение, которое во внешней программе присваивается переменной "y".

Пример №4. Вычислить значения максимального (Max) элемента массива $K(5)$ и минимального (Min) элемента массива $C(9)$. Полученные значения подставить в формулу:

$$N = \frac{(\text{Min} + \text{Max})^2}{\text{Min} + 2}$$

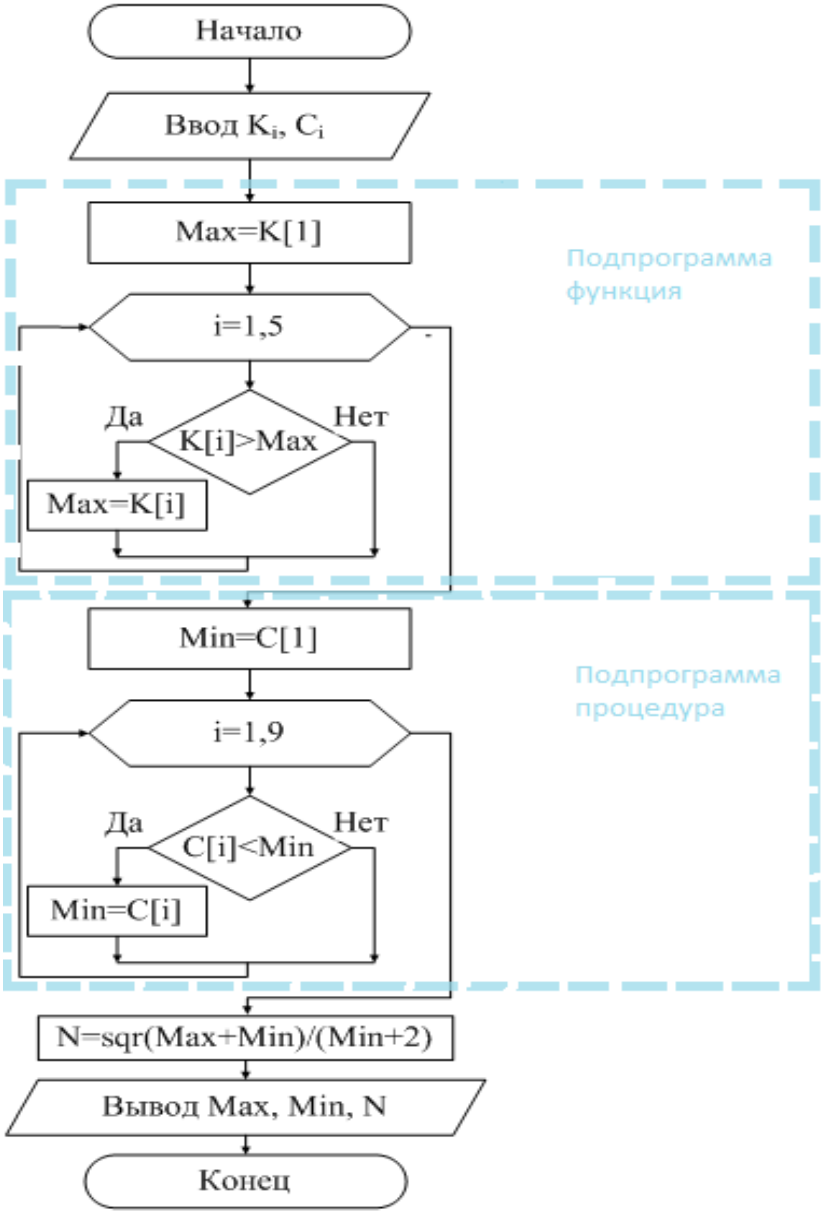
$K(5)$= {7, 80, 25, 72, 31};

$C(9)$= {6.1, 4.2, 3.3, 0.4, –1.5, –8.6, 29.7, 5.8, 17.9}.

Поиск значения максимального элемента оформить в виде подпрограммы функции, поиск значения минимального элемента – в виде подпрограммы процедуры.

Последовательность работы

1. Разрабатываем алгоритм решения задачи. Программирование типовых алгоритмов поиска минимального и максимального элементов. Программирование типовых алгоритмов вычисления суммы и произведения, вычисления факториала, нахождения количества и пр. Минимальный и максимальный элементы массивов вычисляем в разных циклах. Вычисление максимального элемента будем оформлять в виде подпрограммы функции, минимального элемента - в виде подпрограммы процедуры.



Блок-схема алгоритма

2. Составляем программу на языке Паскаль:

```
Program lab_43;
Var                    {Описание переменных параметров}
 N,Min:real; i:integer;
 Type mas1=array[1..5] of integer;
    mas2=array[1..9] of real;
Const                  {Описание постоянных параметров}
 K:mas1=(7, 80, 25, 72, 31);
 C:mas2=(6.1, 4.2, 3.3, 0.4, -1.5, -8.6, 29.7, 5.8, 17.9);
 Function Max(K:mas1):integer; {Подпрограмма функция}
 Var K1,i:integer;     {Вычисление максимального элемента}
Begin
  K1:=K[1];  For i:=1 to 5 do   if K[i]>K1 then K1:=K[i];  Max:=K1;
  end;
   Procedure Minimum(C:mas2;Var Min:real);       {Подпрограмма
процедура}
  Var i:integer; {Вычисление минимального элемента}
Begin
   Min:=C[1];    For i:=1 to 9 do    if C[i]<Min then Min:=C[i];
  end;
 Begin                          {Основная программа}
  Minimum(C,Min);               {Обращение к подпрограмме проце-
дуре}
       N:=sqr(Min+Max(K))/(Min+2);
{Вывод результатов вычислений}
 Writeln('Min=',Min,' Max=', Max(K), ' N=',N:5:2);
End.
```

В приведенном тексте программы, заключенные в фигурные скобки фрагменты являются комментариями (не воспринимаются компилятором). Комментарии используются для пояснения программы и в студенческих работах не обязательны.

3. Входим в среду Pascal ABC, используя иконку на рабочем столе компьютера. Подробно работа в среде Pascal ABC описана в методических указаниях к лабораторной работе № 1.

4. В верхнем окне браузера набираем текст программы на языке Паскаль.

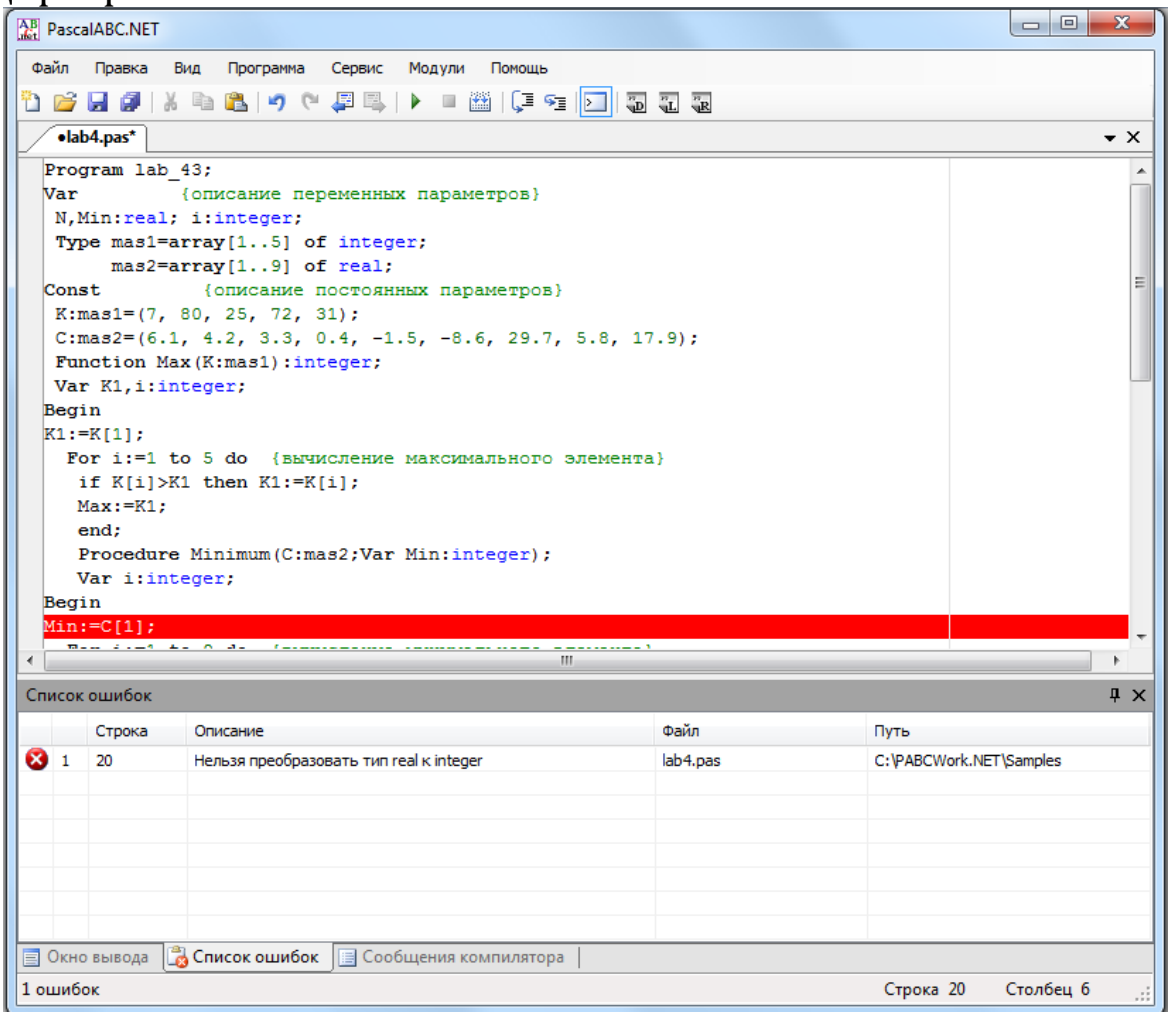
5. Сохраняем текст программы в файле, присвоив ему имя, например ***lab4***.

6. Воспользовавшись иконкой , выполняем программу.

7. При возникновении ошибок, в Окне вывода будет появляться подсказка. Следует отлаживать программу (исправлять ошибки), пока в Окне вывода не появится результат. Каждый раз после исправления ошибки следует сохранять последнюю версию, воспользовавшись иконкой и выполнять программу, активизируя иконку

Разъяснения по наиболее часто встречающимся ошибкам

По большинству возможных ошибок и вариантов их устранения приведены рекомендации в предыдущих примерах. Здесь приведем ошибки, возникающие, в основном, при использовании подпрограмм.

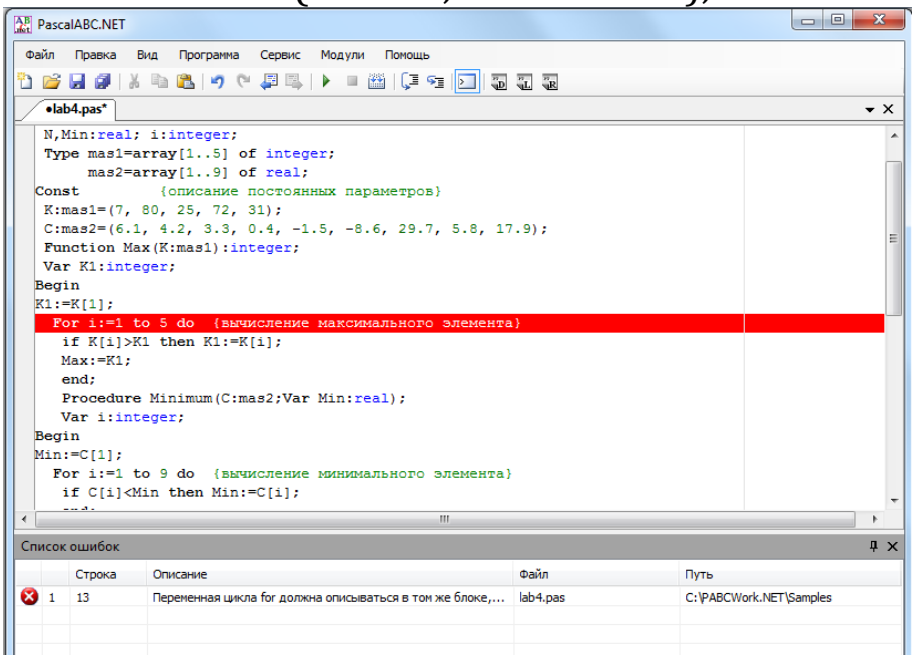


Эта ошибка связана с тем, что в описании глобальных переменных Min описан как вещественная переменная (Min:real;),а в списке формальных параметров процедуры

(**Procedure** Minimum(C:mas2;**Var** Min:integer);) - как целое.

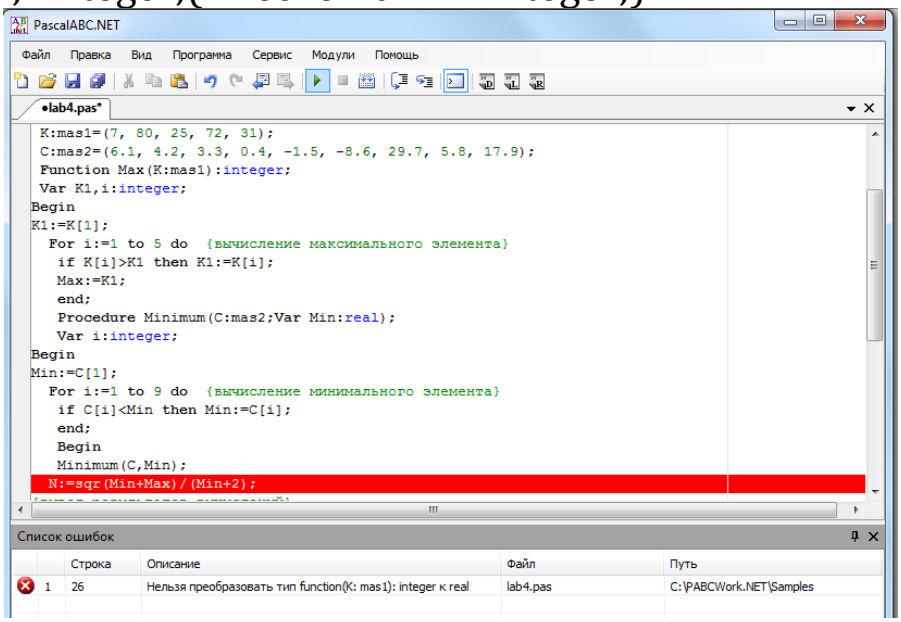
Необходимо исправить описание в списке формальных параметров:

Procedure Minimum(C:mas2;**Var** Min:real);.

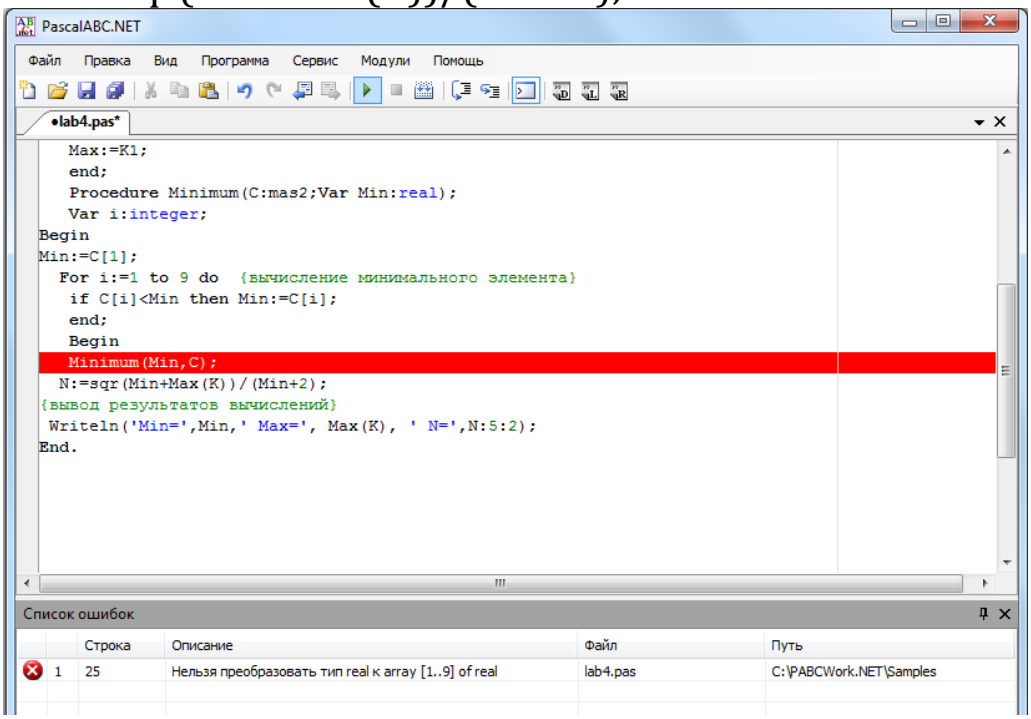


Используемые внутри подпрограммы имена должны быть описаны в списке локальных переменных, в данном случае, переменную цикла For **i** следует внести в список описания локальных переменных:

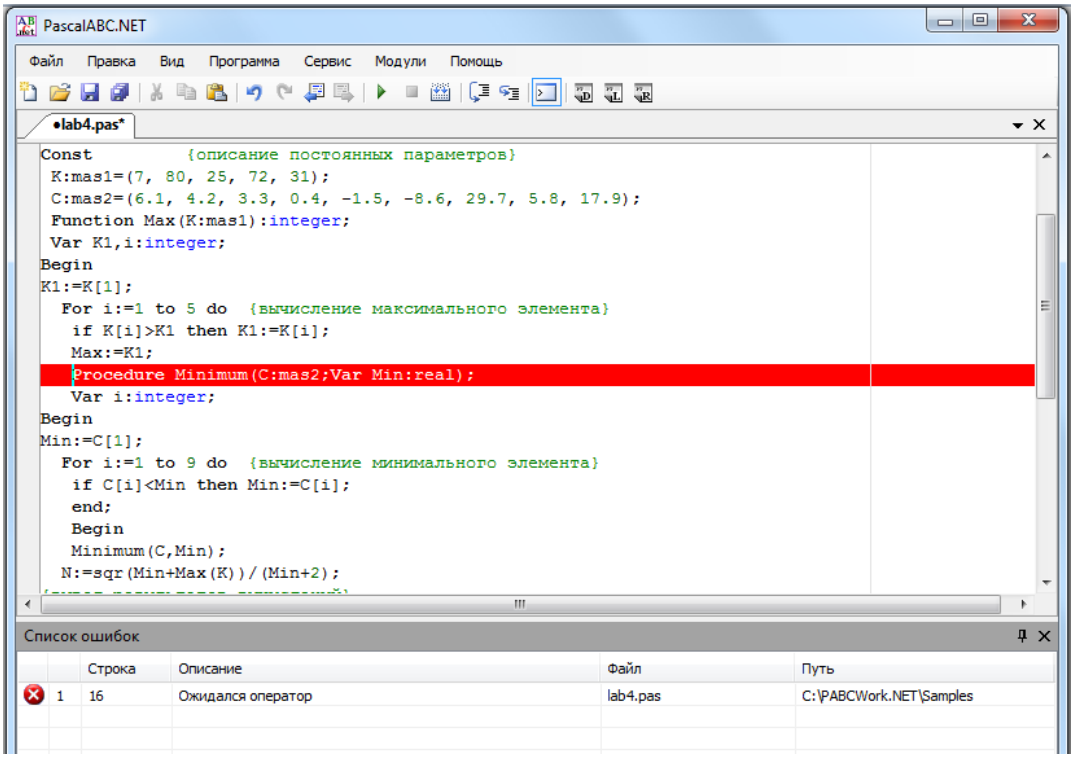
Var K1,i:integer;(вместо **Var** K1:integer;)



В данном случае, Max описан, как подпрограмма функция: (**Function** Max(K:mas1):integer;), поэтому использоваться может только со списком фактических параметров. Т.е. оператор, выделенный красной строкой должен записываться следующим образом: N:=sqr(Min+Max(K))/(Min+2);

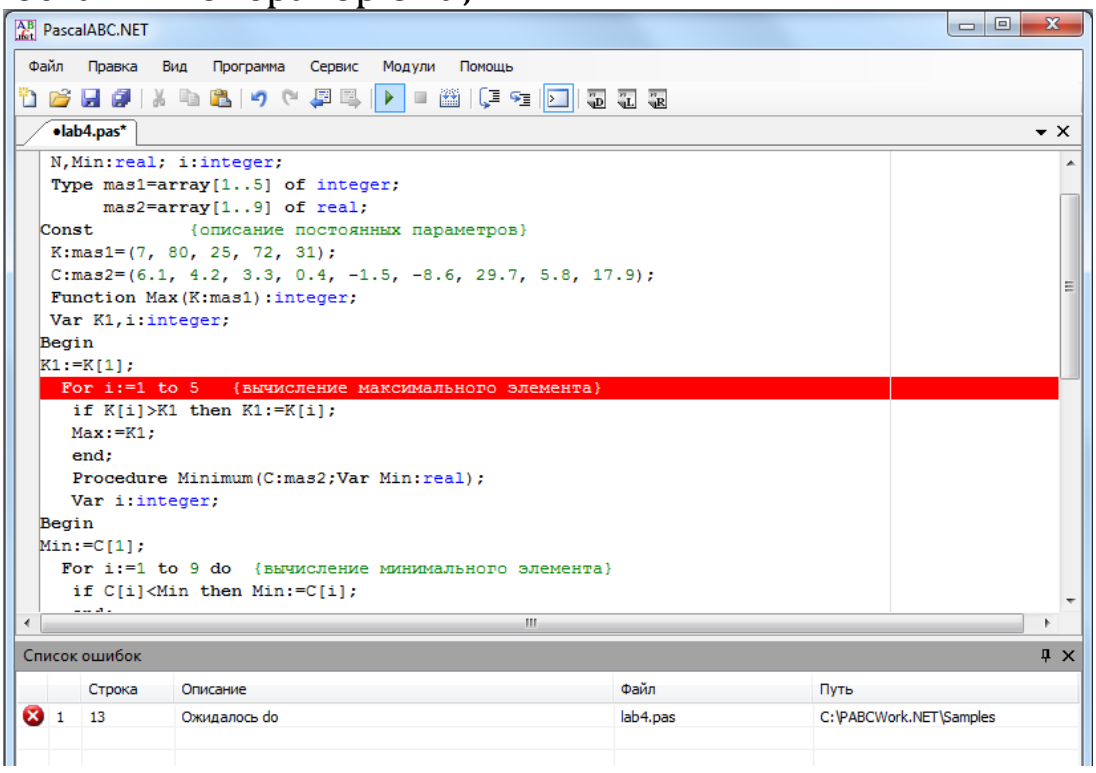


При использовании подпрограммы процедуры (это же касается и подпрограммы функции) между формальными и фактическими параметрами должно быть соответствие по количеству, порядку следования и типу данных. В данном случае, последовательность списка фактических параметров при обращении к подпрограмме процедуре в основной программе Minimum(Min,C) не совпадает с последовательностью списка формальных переменных в заголовке подпрограммы процедуры **Procedure** Minimum(C:mas2;**Var** Min:real). Следует отметить, что имена переменных списков могут не совпадать, а вот соответствие типов переменных списков обязательно диагностируется компилятором и сообщается о возникновении данной ошибки. В нашем случае, обращение к подпрограмме процедуре в основной программе должно быть следующим: Minimum(C,Min).

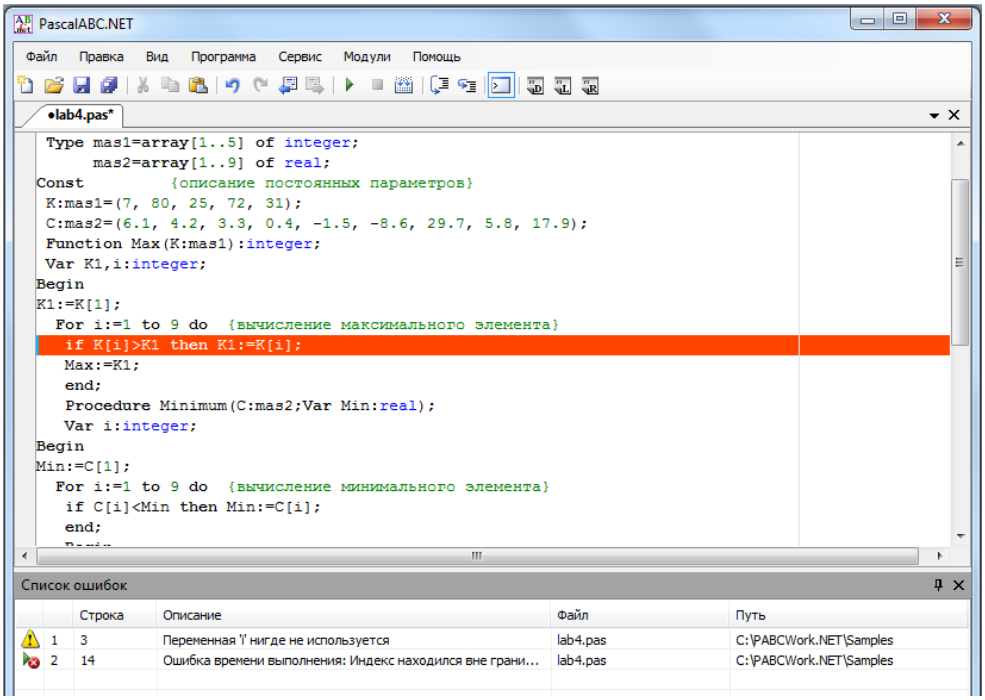


Отсутствует оператор **end**; в конце описания подпрограммы функции, перед оператором, выделенным красной строкой

Procedure Minimum(C:mas2;**Var** Min:real);, являющимся заголовком следующей подпрограммы. Следует перед этим оператором поставить оператор **end**;.

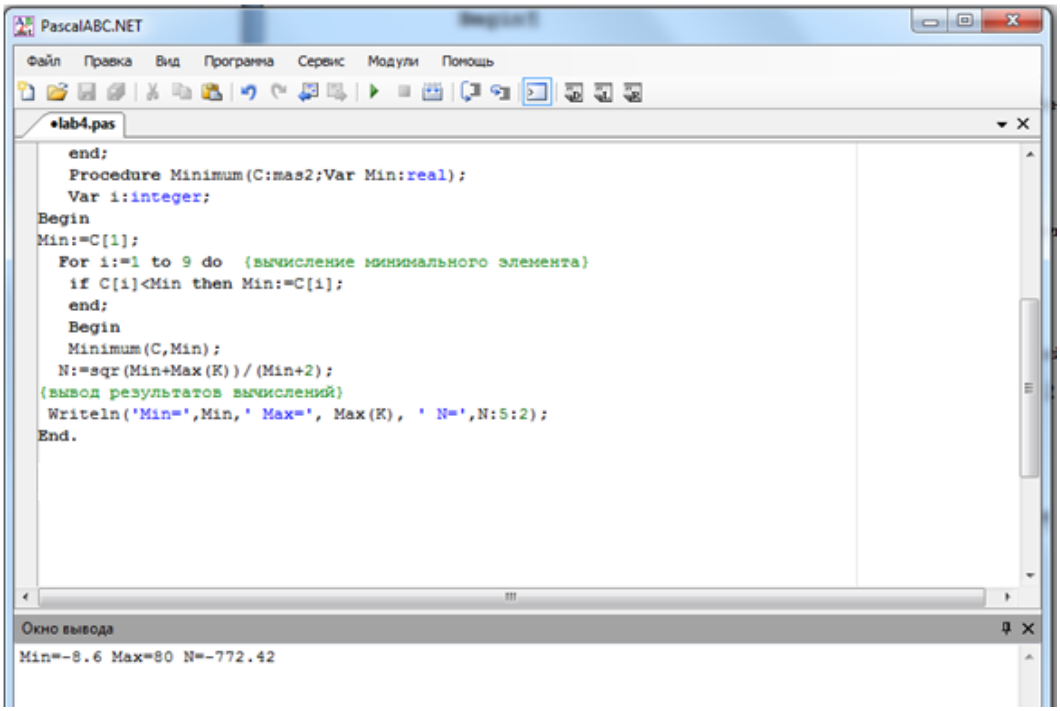


В выделенной строке пропущен оператор цикла **do**. Правильная запись: **For** i:=1 **to** 5 **do**



Верхняя граница цикла (9) выходит за верхнюю границу массива K[i] (5), поэтому и выходит сообщение «…Индекс находится вне границы». В строке, задающей границы циклической процедуры, следует изменить верхнюю границу: **For** i:=1 **to** 5 **do**

Когда программа будет отлажена, в Окне вывода появится результат:



8. Если не удалось завершить отладку программы за один прием, следует записать последнюю версию на диск, и в следующий раз, открыв эту версию продолжить работу (см. пример №1).

Задания для практического занятия

Задание 4-П. Дан массив из десяти целых чисел: {2; −3; 5; 0; 7; −4; 1; −1; −6; 9}. Составить алгоритм, блок-схему и программу на языке Паскаль.

Варианты:

1. Найти сумму четных элементов массива.
2. Найти сумму положительных элементов массива.
3. Найти произведение элементов массива с четными индексами.
4. Найти количество нулевых и количество отрицательных элементов массива.
5. Найти сумму всех элементов массива, которые без остатка делятся на «2».
6. Найти количество положительных элементов массива, не превышающих числа «7».
7. Заменить отрицательные элементы нулями.
8. Увеличить все положительные элементы массива на единицу.
9. Найти произведение ненулевых элементов массива.
10. Увеличить элементы массива с четными индексами на «1», а элементы с нечетными индексами— на «2».
11. Найти сумму отрицательных элементов массива и произведение положительных элементов.
12. Найти среднее арифметическое значение элементов массива.

Задание 5-П. Составить алгоритм, блок-схему и программу на языке Паскаль для вычислений, используя подпрограммы функции и подпрограммы процедуры.

Вырианты

1. Вычислить выражение

$$1\cdot 2 + 2\cdot 3\cdot 4 + 3\cdot 4\cdot 5\cdot 6 + \ldots + n\cdot(n+1)\cdot(n+2)\cdot\ldots 2n$$

2. Элементы *x* и *y* вычисляются по формулам:

$$x_i = 0.3\cdot x_{i-1};\quad y_i = x_{i-1} + y_{i-1},$$

при $x_1 = y_1 = 1.$ Вычислить $\sum_{i=1}^{n} \frac{x_i}{1+|y_i|}$.

3. Вычислить выражение

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^n \cdot \frac{x^{2n}}{(2n)!}$$

4. Вычислить выражение:

$$\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3\cdot 4}+\frac{1}{3\cdot 4\cdot 5\cdot 6}+\ldots+\frac{1}{n\cdot(n+1)\cdot(n+2)\cdot\ldots\cdot 2n}$$

5. Вычислить выражение

$$1\cdot(1+\frac{1}{2})\cdot(1+\frac{1}{2}+\frac{1}{3})\cdot(1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4})\cdot\ldots\cdot(1+\frac{1}{2}+\ldots+\frac{1}{n}).$$

6. Значения членов числовой последовательности a_i, b_i вычисляются по формулам:

$$a_i = 0.8\cdot a_{i-1} + 0.1\cdot b_{i-1};\quad b_i = 0.6\cdot a_{i-1} + 0.2\cdot b_{i-1}, i = 2,3,\ldots$$

Не применяя массивов, вычислить $\sum_{i=1}^{n} a_i$ при $a_1 = b_1 = 1$.

7. Элементы последовательности x_i, $i = 3,4,5,\ldots$ вычисляются по формуле $x_i = 0.8\cdot x_{i-1} + 0.15\cdot x_{i-2}$ при $x_1 = 2$ и $x_2 = 1$. Вычислить $\sum_{i=1}^{n} x_i$.

8. Вычислить сумму $-\frac{2}{4!}+\frac{3}{6!}-\frac{4}{8!}+\ldots+(-1)^{n+1}\cdot\frac{n}{(2n)!}$

9. Значения $\mathbf{x_i}$ вычисляются циклически:

$$x_i = 1.5\cdot x_{i-1} - 0.8\cdot x_{i-2}; i = 2,3,\ldots,k.$$

Вычислить x_k, не применяя массивов, если $x_0 = \cos^2 1; x_1 = -\sin^2 1$.

10. Вычислить выражение: $x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots+(-1)^{n-1}\cdot\frac{x^{2n-1}}{(2n-1)!}$

11. Вычислить выражение $x-\frac{x^3}{1!\cdot 3}+\frac{x^5}{2!\cdot 5}-\frac{x^7}{3!\cdot 7}+\ldots+(-1)^n\cdot\frac{x^{2n+1}}{n!\cdot(2n+1)}$

12. Вычислить произведение первых N сомножителей:

$$\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\ldots$$

13. Числовая последовательность задана формулой

$$x_i = 0.71x_{i-1} + 0.21x_{i-2} + 0.11x_{i-3}, i = 4,5,\ldots.$$

Вычислить x_n, $n \geq 4$, не применяя массивов, если $x_1 = a; x_2 = b;\ x_3 = c$.

14. Вычислить выражение:

$$1-\frac{1}{3}+\frac{2}{5}-\frac{2}{7}+\frac{3}{9}-\frac{3}{11}+\ldots.+(-1)^{n+1}\frac{целое((n+1)/2)}{2n-1}$$

15. Вычислить выражение

$$-\frac{1}{2}+\frac{1}{4}-\frac{2}{6}+\frac{2}{8}-\frac{3}{10}+\frac{3}{12}-\ldots.+(-1)^{n}\frac{целое((n+1)/2)}{2n}$$

16. Вычислить приближенное значение бесконечной суммы

$$\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\frac{1}{3\cdot 4}+\ldots.$$

Нужное приближение считается полученным, если последнее слагаемое, вошедшее в сумму, оказалось меньше данного положительного ε.

17. Вычислить приближенно значение бесконечной суммы

$$\frac{1}{1\cdot 2\cdot 3}+\frac{1}{2\cdot 3\cdot 4}+\frac{1}{3\cdot 4\cdot 5}+\ldots.$$

Нужное приближение считается полученным, если последнее слагаемое, вошедшее в сумму, оказалось меньше данного положительного ε.

18. Числовая последовательность задана формулой $a_i=\frac{2^i}{i!}, i=1,2,\ldots.$ Определить, начиная с какого i, члены последовательности становятся меньше данного положительного числа ε.

19. Числовая последовательность задана формулой $x_i=\frac{i^2}{i!}, i=1,2,\ldots.$ Определить минимальное количество членов k, для которых выполняется условие $\sum_{i=1}^{k} x_i > R,$ где R - заданное число, $R\in(1,\ 5.4365]$.

20. Дано действительное b<0. Последовательность a_1, a_2, ... образуется по следующему закону: $a_1=b;\ a_i=\frac{a_{i-1}+1}{i-\sin^2 i}, i=2,3,\ldots.$ Не используя массивов, найти значение и номер первого неотрицательного члена последовательности.

Задания для самостоятельной работы

Задание № 4-С. Составить алгоритм, программу на языке Паскаль для вычислений, используя подпрограммы функции и подпрограммы процедуры.

1. Члены последовательности вычисляются по формуле $a_i=\frac{1}{i^2};\ i=1,2,\ldots.$ Найти номер i, начиная с которого выполняется условие $a_i-a_{i+1}<\varepsilon$.

2. Сколько членов последовательности $\frac{1}{2},\frac{2}{3},\frac{3}{4},\frac{4}{5},\frac{5}{6},\cdots$ надо просуммировать, чтобы сумма превысила данное значение $S>0$?
3. Дана последовательность: $1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\cdots$ Сколько членов этой последовательности, начиная с первого, и, далее по порядку, надо перемножить, чтобы произведение оказалось меньше данной положительной величины ε?
4. Вычислить приближенное значение бесконечной суммы $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$ Нужное приближение считается полученным, если абсолютное значение последнего слагаемого, вошедшего в сумму, оказалось меньше данного положительного ε.
5. Среди чисел $1,\ 1+\frac{1}{2},\ 1+\frac{1}{2}+\frac{1}{3},\ \cdots$ найти ближайшее меньшее, чем заданное число A.
6. Последовательность чисел формируется по следующему закону: $e_k=\left(1+\frac{1}{k}\right)^k,\ (k=1,2,\ldots).$ Найти номер i ($i\geq 2$) первого члена последовательности, для которого выполняется условие $|e_i-e_{i-1}|<\varepsilon$
7. Элементы последовательности x_i, $i=3,4,5,\ldots$ вычисляются по формуле $x_i=0.8\cdot x_{i-1}+0.15\cdot x_{i-2}$ при $x_1=2$ и $x_2=1$. Вычислить, не применяя массивов, начиная с какого i x_i становится меньше заданного значения Z, ($0<Z<1.3$).
8. Значения членов числовой последовательности a_i, b_i вычисляются по формулам: $a_i=0.8\cdot a_{i-1}+0.1\cdot b_{i-1}$; $b_i=0.6\cdot a_{i-1}+0.2\cdot b_{i-1}$, где $i=2,3,...$; $a_1=b_1=1$. Вычислить, не применяя массивов, начиная с какого i b_i становится меньше заданного значения $S>0$.
9. Последовательность значений имеет вид:
$$1,\cdot 1+\frac{1}{2},\cdot 1+\frac{1}{2}+\frac{1}{3},\cdot 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4},\ldots,\cdot 1+\frac{1}{2}+\ldots+\frac{1}{n}.$$
При каком минимальном n значение произведения членов последовательности от 1-го до n-го становится больше заданного $P>0$?
10. Последовательность значений имеет вид:

$$\frac{1}{1\cdot 2};\frac{1}{2\cdot 3\cdot 4};\frac{1}{3\cdot 4\cdot 5\cdot 6};\ldots;\frac{1}{n\cdot(n+1)\cdot(n+2)\cdot\ldots\cdot 2n}$$

При каком минимальном n значение произведения первых членов последовательности от 1-го до n-го становится меньше заданного $0<P<1$?

11. Рассчитать число зерен, выращенных крестьянином за "N" лет, если он посадил 10 зерен, а годовой урожай составляет 22 зерна на каждое посаженное зерно.
12. Рассчитать число рыбок - самок, выращенных в аквариуме за "N" месяцев, если вначале была одна самка, а ежемесячный прирост от одной самки составляет три самки, причем все рыбки остаются живые.
13. Рассчитать число золотых монет, принесенных в дань господину, если "N+1" подданных последовательно передают монеты от первого к последнему. Причем, первый отдает одну монету, второй увеличивает число монет вдвое, третий - в три раза и т. д.
14. Рассчитать число рыбок, выращенных в аквариуме за "N" лет, если вначале было две рыбки, а число рыбок увеличивается пропорционально числу лет, т. е. 4, 12, 48 и т. д.
15. Рассчитать функцию y=sin(sin(sin(. . . (sin(x))))), в которой имя функции "sin" повторяется N раз.
16. Рассчитать функцию $y=a/(b+(a/(b+(a/(b+(\ldots+a/b))))))$, в которой знак деления "/" повторяется N раз.

Контрольные вопросы

1. Какова структура программы на языке Pascal?
2. Перечислите основные типы данных.
3. Для чего используется оператор присваивания?
4. Каков синтаксис операторов ввода и вывода?
5. Сформулируйте принципы записи сложных выражений в Pascal.
6. Для чего применяется условный оператор if?
7. В каких случаях используют цикл с параметром (цикл FOR)?
8. Сформулируйте принципы использования цикла с предусловием (цикла WHILE).
9. В каких случаях используют цикл с постусловием (цикл REPEAT)?

СПИСОК РЕКОМЕНДУЕМЫХ ИСТОЧНИКОВ

1. Абрамов С.А., Зима Е.В. Начала информатики. - М.: Наука, 1989. - 256 с.
2. Епанешников А.М., Епанешников В.А. Программирование в среде Turbo Pascal 7.0. - М.: Диалог - МИФИ, 1995. - 288 с.
3. Кравцов А.В., Мойзес О.Е., Кузьменко Е.А., Баженов Д.А., Коваль П.И. Информатика и вычислительная математика: учеб. пособие для студентов химических специальностей технических вузов. – Томск: Изд-во ТПУ, 2003. – 245 с.
4. Мойзес О.Е., А.В. Кравцов. Информатика. Ч. 1: учеб. пособие. – Томск: Изд-во ТПУ, 2007. – 126 с.
5. Мойзес О.Е., Баженов, Д.А.Коваль П.И., Кузьменко Е.А. Информатика /Учебное пособие, Томск: Изд. ТПУ, 2000. – 119 с.
6. Овсянкин Е. Ю., Арланова Е. Ю. Основы языка Паскаль. — Самара: РИО СамГТУ. 2010. —76 с.
7. Поляков Д.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль (версия 5.5). М.: - МАИ, 1992. 576 с.
8. Интернет ресурс: http://pascalabc.net.

ОГЛАВЛЕНИЕ

А.АБДИРАШИДОВ,
Б.Б.АМИНОВ,
А.А.АБДУРАШИДОВ.

ПРОГРАММИРОВАНИЕ. ОСНОВЫ РАБОТ В СРЕДЕ Pascal ABC

Методические указания

Muharrir: ***Saydaliyeva N.***

Musahhih: ***Raxmatullayev N.***

Texn. muharrir: ***Ro'ziboyev M.***

2008 yil 19-iyun 68-buyruq.
2016 yil 31-mayda noshirlik bo'limiga qabul qilindi.
2016 yil 18-iyunda original maketdan bosishga ruxsat etildi.
Bichimi 60x84, $^{1}/_{32}$. «Times New Roman» garniturasi.
Ofset qog'ozi. Shartli bosma tabog'i – 4,5.
Nashriyot hisob tabog'i – 3,5.
Adadi 25 nusxa. 49-buyurtma.

SamDU bosmaxonasida chop etildi.
140104, Samarqand sh., Universitet xiyoboni, 15