МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Кубанский государственный
аграрный университет имени И. Т. Трубилина»

Т. А. Анищик

ОСНОВЫ АЛГОРИТМИЧЕСКОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ПАСКАЛЬ
Часть 1

Учебное пособие

Краснодар
КубГАУ
2017

УДК 004.4 (075.8)
ББК 32.973
А67

Р е ц е н з е н т ы :

Е. В. Луценко – профессор кафедры информационных образовательных технологий Кубанского государственного университета, д-р экон. наук, канд. техн. наук, профессор;
В. Н. Лаптев – доцент кафедры компьютерных технологий и систем Кубанского государственного аграрного университета, канд. техн. наук

Анищик Т. А.

А67 Основы алгоритмического программирования на языке Паскаль. Часть 1: учеб. пособие / Анищик Т. А. – Краснодар: КубГАУ, 2017. – 90 с.

Учебное пособие содержит систематизированную справочную информацию и практические задания, которые являются обобщением опыта преподавания в ВУЗе раздела «Программирование на языке Паскаль» курса «Информатики». Контрольные вопросы и задания могут быть использованы при проведении контроля усвоенных знаний: опросов и контрольных работ.

Издание предназначено для студентов-бакалавров агрономических, инженерных, юридических и экономических направлений обучения, изучающих программирование на языке Паскаль.

УДК 004.4 (075.8)
ББК 32.973

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Существует мнение, что будущие специалисты, в том числе и инженеры, не обязаны осваивать курс программирования. Между тем, обучение основам программирования очень важно, поскольку способствует формированию логического мышления, позволяет глубже понять логику работы компьютера, а значит, более полно использовать заложенные в него возможности; и наконец, дает возможность самому создавать и реализовывать программы.

Выбор языка программирования не случаен: язык Паскаль является одним из самых понятных и простых в изучении, содержит средства для создания программ любой сложности. Язык Паскаль изучается в общеобразовательных школах с начала преподавания информатики (1985 г.) и в высших учебных заведениях в качестве одного из основных языков программирования.

Сегодня нет проблемы поиска в сети *Internet* теоретического материала, в том числе и в области программирования. К сожалению, найденное описание бывает либо слишком громоздким и это быстро утомляет пользователя, либо сложно излагается. Кроме этого, еще сложнее найти задания с вариантами для всей учебной группы.

В учебном пособии систематизированное изложение теоретического материала сопровождается примерами программ с пояснениями, что позволит пользователю быстро получить помощь. Для закрепления полученных знаний приведены задания для самостоятельного выполнения и контрольные задания с возможностью сверить результат расчета с приведенным ответом.

В пособии рассмотрены практические вопросы работы с интегрированной средой разработки *Turbo Pascal*, что позволит пользователю, имеющему персональный компьютер (ПК), в сжатые сроки овладеть практическими навыками программирования.

Учебное пособие включает следующие главы:

1. Этапы решения задачи на компьютере. Без изучения этой главы невозможно получить результаты вычислений. Необходимо пройти все этапы разработки решения задачи.

2. Основные элементы языка Паскаль. В главе рассматриваются базовые элементы языка программирования Паскаль: алфавит; простые стандартные и пользовательские типы данных; языковые конструкции, используемые для программирования любых процессов обработки информации.

Ограниченный объем пособия обусловил рассмотрение только алгоритмического программирования на языке Паскаль, т. е. представление программы в виде последовательности модулей.

При описании синтаксических конструкций языка Паскаль использованы некоторые соглашения. Если элемент синтаксического описания заключен:

- в квадратные скобки, то определен необязательный параметр, который может быть опущен без нарушения целостности всей синтаксической конструкции;
- в угловые скобки, то определен обязательный параметр, который нельзя опускать.

Хотелось бы надеяться, что это пособие поможет в нелегком пути освоения программирования на языке Паскаль.

ГЛАВА 1 ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ НА ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

1.1 Основные этапы решения задачи

В процессе получения результатов решения задачи на ПК требуется приложить немало усилий и произвести определенные шаги (этапы). Кроме этого, пользователь должен обладать некоторыми знаниями, чтобы эти этапы пройти. Компьютер может распознавать определенный набор инструкций (***команд***), а затем выполнять их. Последовательность таких инструкций называется ***программой.*** ***Программирование*** – «это процесс создания программистом (человеком) программы (информационной структуры), предназначенной для последующего исполнения (компьютером)» [8].

Рассмотрим основные этапы разработки программы на персональном компьютере:

- ***постановка задачи*** состоит в сборе информации о задаче и четком описании: условий задачи и конечной цели, структуры и типов исходных (начальных) данных, критериев выбора и завершения повторений процессов обработки информации, видов форм выдачи результатов. *Результат*: *спецификация* (набор требований к программному продукту);
- ***исследование задачи*** состоит в анализе технических и программных средств; существующих аналогичных программных продуктов и разработке математической модели, т. е. математической формализации содержания поставленной задачи. *Результат*: *математическая модель* (приближенное описание задачи с помощью математических методов: набора уравнений, неравенств и ограничений и т. д.);
- ***построение алгоритма*** предполагает описание действий в какой-либо форме записи по созданной математической модели. *Результат*: *алгоритм* (точное

описание порядка действий, понятное исполнителю[1], и приводящее исходные данные к конечному результату);

- ***программирование*** – это этап составления текста программы по разработанному алгоритму с использованием команд выбранного *языка программирования* (система условных обозначений для записи команд). Этапы разработки программы возможны в среде программирования, связанной с выбранным языком программирования. *Результат*: программный код;
- ***отладка и тестирование программы*** состоит в выявлении и устранении ошибок и причин их появления. Тестирование программы заключается в выявлении *синтаксических ошибок* (ошибки в записи конструкций языка программирования) и *семантических ошибок* (ошибки, связанные с неправильным содержанием действий и использованием недопустимых значений величин). Для выполнения тестирования программы подбирают наборы значений входных данных с заранее известными результатами (*тесты*) или могут быть легко просчитаны. *Результат*: программа;
- ***выполнение расчетов с реальными данными*** возможно, если выполнены все этапы разработки. *Результат*: замечания;
- ***анализ полученных результатов***. Если обнаруживаются ошибки, то вполне возможен возврат к любому из этапов;
- ***сопровождение программы*** включает доработку программы и составление сопроводительной документации к программе. *Результат*: документированное приложение.

[1] Будем считать, что исполнителем алгоритма является вычислительное устройство, которое распознает описания алгоритмов.

1.2 Основы работы в интегрированной среде разработки *Turbo Pascal*

Этап реализации созданной программы на ПК возможен в *интегрированной среде разработки*. Интегрированная среда разработки *Turbo Pascal* состоит из компилятора и инструментальной программной оболочки [1].

Компилятор – это программа, которая преобразует программный код, сохраненный в текстовом файле **.pas* в двоичный код с созданием двоичного файла – **.exe*.

Инструментальная программная оболочка содержит все необходимые средства для создания программ: редактор программ, средства автоматизации сборки и отладчик.

После запуска *Turbo Pascal (TP)* на экране открывается окно, содержащее три области интегрированной среды: рабочая область, меню и строка состояния.

В *рабочей области* открывается окно для набора и редактирования программы (рисунок 1).

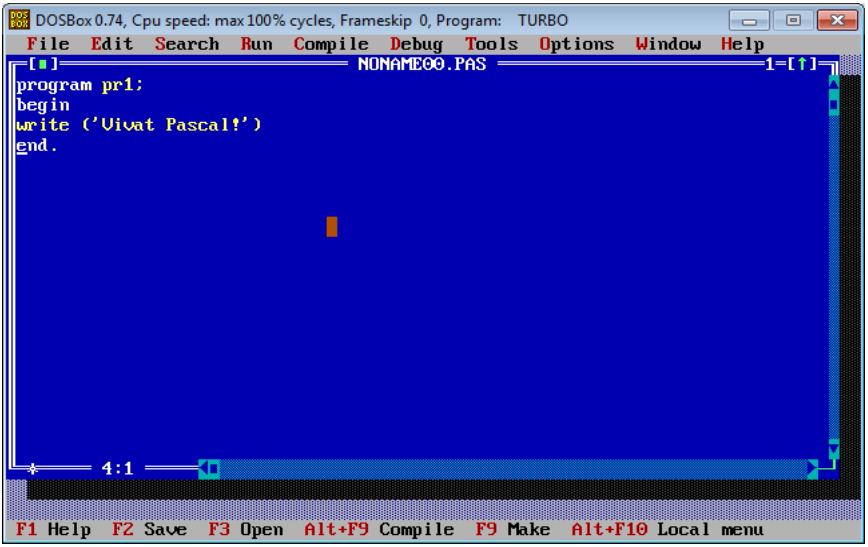

Рисунок 1 – Вид окна *Turbo Pascal*

Строка меню содержит следующие пункты:

- ***File*** (файл). Содержит набор команд с файлами и папками. Подменю содержит следующие пункты:
  – *New* открывает новое окно экрана и создает новый файл с именем *NONAME(номер).PAS;*
  – *Open…* в диалоговом окне запрашивает и потом загружает в окно редактора файл с расширением **.pas*. Можно ввести имя, которого нет в текущем каталоге, тогда будет создан новый файл;
  – *Save* сохраняет файл, ранее сохраненный, иначе среда запросит новое имя для *NONAME(номер).pas;*
  – *Save as…* в диалоговом окне запрашивает и потом сохраняет файл под новым именем;
  – *Save all* сохраняет содержимое всех окон редактора в виде файлов;
  – *Change dir…* в диалоговом окне запрашивает и потом изменяет текущий каталог пользователя;
  – *Print* выводит текст программы из окна на принтер;
  – *Printer setup…* позволяет выполнять настройки принтера и параметры печати;
  – *Dos shell* осуществляет временный выход в *DOS*. Для возврата в среду используется команда *Exit*;
  – *Exit* осуществляет выход их среды *Turbo Pascal;*
- ***Edit*** (редактирование). Включает все основные операции для редактирования текста. Подменю содержит следующие пункты:
  – *Undo* отменяет предыдущее действие;
  – *Redo* отменяет действие команды *Undo*;
  – *Cut* вырезает выделенный блок текста в буфер обмена;
  – *Copy* копирует выделенный блок текста в буфер обмена;
  – *Paste* вставка содержимого буфера обмена в позицию курсора;

– *Clear* удаляет выделенный блок без помещения его в буфер обмена;
– *Show clipboard* отображает содержимое буфера обмена;

- ***Search*** (поиск). Предоставляет возможность осуществлять поиск и замену фрагментов текста. Подменю содержит пункты:
  – *Find…* в диалоговом окне запрашивает фрагмент текста для поиска и потом осуществляет поиск;
  – *Replase…* в диалоговом окне запрашивает фрагмент текста для поиска и замены на новый фрагмент. Если включена опция *Change all*, то замены производятся автоматически без запросов на подтверждение;
  – *Search again* осуществляет повторение поиска;
  – *Go to line number...* в диалоговом окне запрашивает номер строки для быстрого перехода к строке;
- ***Run*** (запуск). Предназначен для запуска программы, в том числе в пошаговом режиме. Подменю содержит следующие пункты:
  – *Run* запускает операции компиляции, компоновки и выполнения программы автоматически;
  – *Step over* запускает режим построчного выполнения программы (*трассировка*) без подпрограмм;
  – *Trace Into* запускает режим построчного выполнения программы, включая подпрограммы;
  – *Go to cursor* осуществляет отладку фрагмента программы от выделенной голубым цветом строки до строки, в которой находится курсор;
  – *Program reset* прерывание режима трассировки;
- ***Compile*** (компилирование). Осуществляет компиляцию программы. Подменю содержит пункты:
  – *Compile* компилирует текущую программу или из файла, указанного в *Primary file*;

– *Make* компилирует программу вместе с измененными модулями пользователя;
– *Build* компилирует программу вместе со всеми модулями пользователя;

- ***Debug*** (отладка). Содержит команды, которые определяют и изменяют значения переменных в ходе трассировки, просматривают содержимое стека обращений к процедурам, позволяют проводить эффективную отладку программы и т. д. Подменю содержит пункты:
  – *Breakpoints…* в диалоговом окне позволяет устанавливать контрольные точки останова программы с целью выявления ошибок;
  – *Call Stack* показывает последовательность процедур, вызванных программой перед запуском выполняющейся процедуры;
  – *Register* показывает состояние регистров микропроцессора и используется при отладке встроенных модулей;
  – *Watch* открывает окно отладки и ждет ввода выражения;
  – *Output* позволяет во время отладки программы просматривать исходный текст, переменные и вывод одновременно;
  – *User Screen* позволяет просматривать в полноэкранном режиме вывод результатов работы программы;
  – *Evaluate/Modify* позволяет оценить переменную или выражение, просмотреть значение любой переменной или другого элемента данных и изменить значение простых элементов данных;
- ***Tools*** (инструменты). Включает в себя некоторые дополнительные средства *Turbo Pascal*;
  – *Messages* открывает окно сообщений, в которое программа помещает полученные результаты, если она выводит информацию через фильтр *DOS*;

- *Go to next* позволяет переходить к следующей строке окна сообщений;
- *Go to previous* позволяет переходить к предыдущей строке окна сообщений;
- *GREP* (утилита), которая осуществляет поиск информации в указанных файлах с размещением результата в окне сообщений;

- ***Options*** (опции). Устанавливает параметры компилятора и среды разработчика. Подменю содержит следующие пункты:
  - *Compiler…* в диалоговом окне позволяет выбирать тип сгенерированного кода, тип сообщений об ошибках во время выполнения программы, количество компилируемой информации об отладке и т. д.;
  - *Memory sizes…* в диалоговом окне позволяет изменять заданные по умолчанию требования памяти для программы;
  - *Linker…* в диалоговом окне позволяет определить, как компоновщик будет компоновать программу;
  - *Debugger…* в диалоговом окне находятся несколько установок, которые влияют на работу интегрированного отладчика;
  - *Directories…* в диалоговом окне позволяет определить каталоги, которые используются при выполнении и при сохранении программы;
  - *Tools…* в диалоговом окне позволяет добавлять или удалять программы, запускать другую программу без выхода из среды *Turbo Pascal*;
  - *Environment* открывает подменю, из которого можно вызвать шесть диалоговых окон, с помощью которых можно настраивать среду *Turbo Pascal*;
  - ***Window*** (окно). Содержит все основные операции с окнами. Подменю содержит следующие пункты:

– *Tile* меняет расположение окон на экране: видны все окна и имеют одинаковые размеры;
– *Cascade* меняет расположение окон на экране: окна располагаются каскадом, перекрывая друг друга;
– *Zoom* меняет вид представления окна на экране: распахивает окно на весь экран или возвращает ему прежний вид;
– *Next* делает активным следующее открытое окно;
– *Close* закрывает активное окно;

- ***Help*** (справка). Предоставляет справочную информацию. Подменю содержит следующие пункты:
  – *Contents* содержит все главные темы справочной системы;
  – *Topic Search* содержит полные пояснения к оператору или служебному слову, под которым в данный момент находится курсор;
  – *Files…* в диалоговом окне позволяет устанавливать или удалять дополнительные справочные файлы;
  – *Standart units* выдает список стандартных модулей;
  – *Turbo Pascal Language* выдает список элементов языка *Pascal*.

При выборе пунктов, вертикальных меню может появиться диалоговое окно, в котором уточняются выбранные действия. В окне может быть несколько полей (переход *<Tab>).*

Строка состояния содержит информацию о важных в данный момент операциях или соответствующие им комбинации клавиш.

1.3 Этапы разработки программы в среде *Turbo Pascal*

Разработка программы в среде *Turbo Pascal* проходит несколько этапов (таблица 1). По понятным причинам, одни и те же программы, написанные разными пользователями, могут существенно различаться. Бывает, что правильно работающие программы содержат такой программный код, что разобраться

и внести изменения довольно сложно и долго. В этом случае, программисты говорят, что проще написать новую программу, чем разбираться в такой программе.

Таблица 1 – Этапы разработки программы

Номер и название этапа	*Выполнение*
1 этап: создание	*<File> – <New>* или *<File> – <Open> – имя файла*
2 этап: сохранить как…	*<File> – <Save as…>*
3 этап: компиляция	*<Compile> – <Compile>* или *<ALT> + <F9>*
4 этап: выполнение	*<Run> – <Run>* или *<CTRL> +<F9>*
5 этап: просмотр результатов	*<Debug> – <User screen>* или *<ALT> + <F5>* или *<Debug> – <Output>*
6 этап: возврат в среду *TP*	*<Exit>* или *<Esc>*
7 этап: выход из *TP*	*<File> – <Exit >* или *<ALT> + <x>*

Назначение функциональных клавиш представлено в таблице 2.

Соблюдение общих принципов создания программ не гарантирует написание безошибочных программ, но позволяет минимизировать ошибки:

– команды, соответствующие шагам в разработанном и описанном алгоритме, выполняются сверху вниз: последовательно, либо по одной из ветвей, либо циклически;

– для выполнения программы необходимы данные, которые могут быть получены путем присваивания результатов вычисления, при вводе с клавиатуры, из файла или как результат работы другой программы;

– все данные, которые используются программой, должны быть описаны;

– в программе должны быть определены все точные критерии для выполнения тех или иных условий;

– циклические процессы, описанные в программе, должны иметь условие выполнения или условие завершения, иначе происходит «зацикливание» программы;

– программа должна иметь результат, который выводится на экран дисплея или печатающее устройство.

Таблица 2 - Назначение функциональных клавиш

Название клавиши	*Пояснение*
<F1>	– вызов справки
<F2>	– повторное сохранение текста программы
<F3>	– открытие текста программы
<F4>	– выполнение программы от начала до строки, на которой расположен курсор
<F5>	– окно редактора раскрывается на весь экран (повторное нажатие возвращает исходное изображение)
<F6>	– меняет окно редактора *Edit* на окно отладки *Watch*
<F7>	– трассировка программы
<F8>	– то же, что и *F7*, но при выполнении процедуры последняя будет выполняться за один шаг, как строка
<F9>	– компиляция программы
<F10>	– вызов главного меню *Turbo Pascal.*

Если следовать принципам *структурного программирования*[2], сформулированных Э. Дейкстрой [5], то можно существенно сократить количество ошибок и упростить внесение изменений (*модификация*) в программу:

[2] *Структурное программирование* основано на использовании типовых управляющих структур алгоритмов обработки данных.

– отдавайте предпочтение более простым методам программирования более сложным;

– не следует создавать большие по объему программы – разбивайте по возможности программу на логически завершенные блоки;

– безусловная передача управления (оператор *goto*) запрещена, поскольку приводит к нарушению принципа: каждая сложная команда в программе должна иметь одну точку входа и одну точку выхода. Об использовании этого оператора писал Э. Дейкстра в статье «О вреде оператора *GOTO*»;

– повторяющиеся фрагменты *основной* (главной) программы можно оформить в виде ***подпрограмм***;

– использование в любой программе трех базовых управляющих конструкций: последовательность, ветвление и цикл.

Упражнения для самостоятельного выполнения

Упражнение 1. Выполните следующие действия:

- запустите среду *Turbo Pascal*;
- выполните 1 этап (см. таблица 1) через команду из главного меню *TP*: <*File*> – <*New*>;
- наберите в окне редактора программу:

```
program pr1; begin write ('Vivat Pascal!') end.
```

- выполните 2 этап (см. таблица 1): сохраните программу командой из главного меню *TP*: <*File*> – <*Save – as,,,*> в файле с именем *prog1*;
- выполните 3-4 этапы;
- выполните 5 этап: посмотрите результат выполнения программы командой из *TP*: <*Debug*> – <*User screen*>;
- выполните 6 этап;
- закройте файл *prog1* (щелчком левой кнопки мыши по квадратику в левом верхнем углу рабочего поля экрана).

Пояснения к программе:

Компилятор воспринимает программу потоком, но даже такая простая программа сложно воспринимается пользователем, поэтому в программе принято делать отступы для ее удобочитаемости (см. рисунок 1).

Упражнение 2. Выполните следующие действия:

- выполните 1 этап (см. таблица 1) через команду из главного меню *TP*: <*File*> – <*Open*> с именем *prog2*;
- установите режим просмотра результата выполнения программы командой из главного меню *TP*: <*Debug*> – <*Output*>;
- наберите текст программы:

```
program pr2;
var x, y, p: real;
  begin
        write (‘ x=, y= ’);
        readln (x, y);
        p:= x * y;
        writeln (‘ p= ’, s)
  end.
```

- сохраните программу (<*F2*>) (см. таблица 2);
- выполните 3-4 и 6 этапы;
- закройте файл *prog2*.

Пояснения к программе:

Открытие несуществующего файла с именем *prog2* приведет к его созданию. После ввода значений двух переменных (*x, y*), программа вычислит произведение чисел и выведет результат в отдельное окно внизу экрана.

Упражнение 3. Выполните следующие действия:

- выполните 1 этап (см. таблица 1) через команду из главного меню *TP*: <*File*> – <*Open*> с именем *prog1*;
- внесите изменения в программу:

```
program pr3;
var d, m, g: integer;
  begin
```

```
        writeln (‘ Vivat Pascal!’)
        write (‘ Data: dd.mm’);
        readln (d, m);
        writeln (‘ Today: ’, d, ‘.’, m);
 end.
```

- выполните 2 этап: сохраните программу в файле с именем *prog3*;
- выполните 3-4 этапы;
- выполните 5 этап: посмотрите результат выполнения программы через набор комбинации клавиш: <*Alt*> + <*F5*>;
- выполните 6 этап;
- дополните текст программы вводом значения текущего года:

```
   write (‘ Data: gggg’);
   readln (g);
   writeln (‘ Today: ’, d, ‘.’, m, ‘.’, g)
```

- сохраните измененную программу (<*F2*>) (см. таблица 2);
- выполните 3-6 этапы;
- закройте файл *prog3*.

Пояснения к программе:

В программе определены инструкции с запросом даты. Необходимо ответить программе на запрос и ввести данные через <Пробел> или <*Enter*>, иначе программа будет ожидать ответ. Программа выводит на экран сформированную дату.

Упражнение 4. Выполните следующие действия:

- откройте файл *prog3* (<*F3*>) (см. таблица 2);
- внесите изменения в предыдущую программу:

```
program pr4;
uses Crt;
var d, m, g: integer;
 begin
        ClrScr;
        TextBackGround (Lightgreen);
        TextColor (Black);
```

– скопируйте две предыдущие строки в буфер обмена командой из главного меню *TP*: <*Edit*> – <*Copy*>;

```
write (‘ Data: dd.mm’);
readln (d, m);
```

– вставьте строки из буфера обмена командой: <*Edit*> – <*Paste*> и внесите изменения:

– *Lightgreen* замените на *Magenta*;

– *Black* замените на *White*;

```
writeln (‘ Today: ’, d, ‘.’, m);
```

– вставьте строки из буфера обмена командой: <*Edit*> – <*Paste*> и внесите изменение:

– *Lightgreen* замените на *Blue*;

```
write (‘ Data: gggg’);
readln (g);
```

– вставьте строки из буфера обмена командой: <*Edit*> – <*Paste*> и внесите изменение:

– *Lightgreen* замените на *Red*;

– *Black* замените на *White*;

```
writeln (‘ Today: ’, d, ‘.’, m, ‘.’, g)
```

end.

– сохраните программу в файле с именем *prog4*;

– выполните 3-6 этапы;

– выполните трассировку программы (<*F7*>) (см. таблица 2).

Пояснения к программе:

Программа дополнена специальными процедурами модуля *Crt*: очистки экрана, изменения цветов фона и символов.

Упражнение 5. Выполните следующие действия:

– откройте ранее созданные файлы с именами *prog1 – prog4*;

– смените режимы просмотра расположения окон на экране командами из главного меню *TP* поочередно:

<*Window*> – <*Tile*>; <*Window*> – < *Cascade*>;

<*Window*> – < *Next*>;<*Window*> – < *Zoom*>;

– закройте сразу все открытые окна командой из главного меню *TP*: <*Window*> – < *Close all*>.

Контрольные вопросы к главе 1

1.	Какие этапы нужны для решения задачи на ПК?
2.	Каково назначение пункта <*Edit*> из меню *TP*?
3.	В чем состоит этап постановки задачи?
4.	Чем принципиально отличаются языки программирования от естественных языков?
5.	Каково назначение пункта <*File*> из меню *TP*?
6.	В чем состоит этап построения алгоритма?
7.	По каким принципам создаются программы?
8.	Каково назначение пункта <*Compile*> из меню *TP*?
9.	Какие существуют способы описания алгоритмов?
10.	Чем определяется язык программирования?
11.	Каково назначение пункта <*Help*> из меню *TP*?
12.	В чем состоит этап построения математической модели?
13.	По каким критериям выбирают язык программирования?
14.	В чем состоят преимущества процесса компиляции?
15.	Каково назначение пункта <*Run*> из меню *TP*?
16.	Что такое алгоритм? Какие бывают алгоритмы?
17.	Что представляет собой язык программирования?
18.	Что включает в себя интегрированная среда разработки?
19.	Какие существуют критерии построения алгоритмов?
20.	Каково назначение пункта <*Debug*> из меню *TP*?
21.	Что такое инструментальная программная оболочка?
22.	Как можно классифицировать языки программирования?
23.	Каково назначение пункта <*Window*> из меню *TP*?
24.	Какие существуют свойства алгоритмов?
25.	В чем состоит принцип «умолчаний и соглашений»?
26.	Какие существуют этапы разработки программы?
27.	В чем состоит отличие этапов отладки и тестирования?
28.	Каково назначение пункта <*Options*> из меню *TP*?
29.	В чем отличие синтаксических и семантических ошибок?

ГЛАВА 2 ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

2.1 Алфавит языка Паскаль

Считается, что ***алфавит языка программирования*** – это конечный, фиксированный набор символов, используемых для составления текстов на данном языке. Совокупность латинских букв (прописные и строчные), арабские цифры от 0 до 9 и специальные символы образуют *алфавит языка Паскаль*. Из символов, входящих в алфавит языка, и составляют *слова*, отделяемые друг от друга *разделителями* (пробелы, любые управляющие символы и комментарии).

Знаки пунктуации, знаки операций и некоторые *зарезервированные слова*[3] образуют *специальные символы* языка, которые представляют собой одиночные или двойные (составные) символы, которые нельзя разделять пробелом (приложение 1). К зарезервированным словам, например, относятся имена команд, имена некоторых логических и арифметических функций. Список зарезервированных слов языка Паскаль приводится в приложении 2.

Буквы русского алфавита и некоторые символы (&, %) могут быть использованы в символьных строках, а также в программах в виде комментария, обрамляемого фигурными скобками. При компиляции программы комментарии пользователя игнорируются.

Существуют два способа задания директив (указаний) компилятору: программно: {$ <имя директивы>} и через систему вложенных меню интегрированной среды *Turbo Pascal:* <*Options*> <*Compiler*> (см. гл. 1 п. 1.3).

---

[3] *Зарезервированные слова* – это слова, написание и использование которых определил автор языка программирования. Все зарезервированные слова языка Паскаль в интегрированной среде выделяются белым цветом; в дальнейшем изложении будем выделять зарезервированные слова курсивом. Например: *begin*, *type*.

2.2 Понятие идентификатора

Идентификаторы строятся из букв, цифр и знаков подчеркивания, начиная с буквы, и служат для обозначения (именования) переменных, констант и некоторых других объектов.

Существуют *стандартные* идентификаторы для обозначения имен подпрограмм, имен простых типов данных и т. д. Идентификаторы может создавать и сам пользователь. В этом случае, идентификаторы называются *пользовательскими*.

Требования к созданию:

– написание прописных и строчных букв в идентификаторах не различается, например, слова *Massiv, massiv* и *MASSIV* трактуются как одно слово;

– желательно давать переменным не очень длинные «говорящие» имена, например, *sum, N, h, R* и т. д.;

– имя не должно начинаться с цифры;

– недопустимо в идентификаторах использование пробелов, знака тире, точки;

– длина идентификатора может быть до 255 символов, но учитываются только первые 63 символа, т. е. если имена совпадают в 63 символах, то они считаются неразличимыми;

– одно и то же имя переменной нежелательно использовать под хранение разных величин;

– в качестве идентификаторов запрещено использовать зарезервированные слова;

– запрещено использование в качестве имен стандартные идентификаторы, например, имена встроенных констант и подпрограмм.

Примеры неправильных идентификаторов:

1_Index – имя начинается с цифры;

kol vo – использование пробела;

Prog-1 – использование тире;

Stroka.2 – использование точки;

Type – использование зарезервированного слова.

2.3 Состав и структура типов данных

В любой момент времени выполнения программы, данные имеют *имя* (идентификатор), какое-то *значение*, либо не определены. Например, если значения данных представляют собой символы, то говорят, что это символьные данные и т. д. Набор допустимых значений, которые могут принимать данные, и определяют их *тип*. Типы данных связаны с допустимыми операциями.

Константами (*constant*) называются данные, не изменяющие своего значения при выполнении программы, в которой они определены.

Описание: *const* <идентификатор> = <значение константы>;

Пример: пусть описаны следующие константы:

```
const   h = 1.15;
        N = 10;
        otv = 'Y';
        name = 'Язык Паскаль';
```

По значению константы распознается ее тип автоматически без предварительного описания:

– если в числе присутствует точка, то это вещественный тип (*h*), иначе целый тип (*N*);

– если присутствует одиночный символ в апострофах[4], то это символьная константа (*otv*), иначе строковая константа (*name*).

Существуют зарезервированные константы, т. е. к ним можно обращаться без описания. Например, *pi* – «число π», *true* – «истина», *false* – «ложь».

Если описать значение как константу, то можно использовать это имя в программе сколько угодно раз с неизменным содержанием. Если возникает необходимость уточнить значение константы, то это проделывается один раз в разделе описаний. При выполнении программы компилятор каждый раз

[4] *Апостроф* – знак в виде надстрочной запятой (’), набираемый с клавиатуры: англ. алфавит, клавиша с буквой <э>.

встречая идентификатор константы, замещает его значением.

Переменными (*variable*) называются данные, значения которых изменяются в процессе выполнения программы.

Описание: *var* < идентификатор>:<тип>;

Описание нескольких однотипных переменных $a_1, a_2, ..., a_n$ можно оформить так:

var $a_1, a_2, ..., a_n$: <тип>;

а описание разнотипных переменных можно объединить по типам следующим образом:

var $a_1, a_2, ..., a_n$: <тип 1>; … $w_1, w_2, ..., w_к$: <тип N>;

Все используемые типы данных в языке Паскаль подразделяются на группы: простые, структурированные, ссылочные (указатели), процедурные и объекты.

В программах можно использовать как стандартные типы данных, так и типы данных, определяемые пользователем. Стандартные типы данных не требуют предварительного описания, к ним относятся:

- целые типы: *integer, byte, shortint, word, longint*;
- вещественные типы: *real, single, double, extended, comp*;
- логический тип *boolean;*
- символьный тип *char;*
- *ASCII*-строка *pchar;*
- строковый тип *strin*g;
- текстовый файл *text;*
- тип указатель *pointer*.

Все остальные типы данных должны быть определены либо в разделе объявления типов, либо в разделе объявления переменных или констант; к ним относятся:

- регулярный тип *array*;
- комбинированный тип *record*;
- двоичный файл *file;*
- множественный тип *set;*
- процедурный тип *procedure;*
- процедурный тип *function;*
- тип объект *object.*

В общем случае определение типа данных имеет вид:

type

<имя типа 1> = <определение типа 1>;

<имя типа 2> = <определение типа 2>;

……

<имя типа *N*> = <определение типа *N*>;

Описание переменной определенного таким образом типа задается обычным образом: *var* <имя переменной>:<тип>;

Возможен упрощенный вариант описания переменных без предварительного описания типа, так называемое *правило совмещения описаний* типа данных и переменных этого типа:

var <имя переменной>:<определение типа>;.

2.4 Простые стандартные типы данных

В любой момент времени выполнения программы переменные простых или скалярных (*scalar* – простой) типов принимают только одно значение, которое не имеет выраженной структуры.

К простым стандартным типам данных относятся [2]:

- ***целый тип*** *(integer).* Данные этого типа принимают целочисленные значения.

Описание: <идентификатор>: integer;

Пример: *var* a, b: integer;

Операции: +, -, *, /, div, mod.

Стандартные функции, результат которых – целое число, представлены в приложении 3.

Диапазон принимаемых значений данных целого типа: -32768..+32767.

Для хранения целочисленного данного типа integer выделяется 2 байта памяти. Типы целочисленных данных: *byte* (1 байт) – короткое положительное целое в диапазоне значений: 0..255, *shortint* (1 байт) – короткое целое в диапазоне: 128..127, *word* (2 байта) – короткое положительное целое в диапазоне: 0..65535, и *longint* (4 байта) – длинное целое для

размещения больших чисел в диапазоне: 2147483648..2147483647;

- ***вещественный тип*** *(real)*. Вещественные десятичные числа могут быть представлены в двух форматах: либо с фиксированной точкой, либо с плавающей точкой.

В первом формате представления целая часть отделяется от дробной части точкой. Если число целое, то точка отсутствует. Перед числом может указываться знак плюс (+) или отсутствие знака, если число положительное; минус (-), если отрицательное число.

Вещественные числа с плавающей точкой представляются в экспоненциальном виде*:* $mE+p$, где m – мантисса, «*E*» означает «10 в степени», p – порядок.

Пример: 100.1E-01 = $100.1*10^{-1}$= 10.01

Результат выполнения операций над данными этого типа, скорее всего, будет с погрешностью, величина которой определяется количеством разрядов, используемых для представления чисел с плавающей точкой.

Описание: <идентификатор>: real;

Пример: *var* x, y: real;

Операции: +, -,*, /, дают в результате вещественное число.

Стандартные функции, результат которых – вещественное число, представлены в приложении 3.

Диапазон принимаемых значений данных вещественного типа $2.9*10^{-39}$ ..$1.7*10^{38}$.

Для хранения вещественного данного типа *real* компилятор выделяет 6 байт памяти. Типы вещественных данных: *single* (4 байта), *double* (8 байт), *comp* (8 байт), *extended* (10 байтов);

- ***символьный тип*** *(char).* Данные принимают значения из набора *ASCII*-символов. Символьные значения обязательно указываются в апострофах. Например: ’*’, ’ ’, ’Ж’, ’0’.

Описание: <идентификатор>: char;

Пример: *var* s1, s2: char;

Стандартные функции работы с символьными переменными представлены в приложении 3.

Значение символьных данных можно сравнивать. На самом деле, происходит сравнение кодов символов из таблицы кодировки *ASCII*.

Для хранения одного символа потребуется 1 байт.

- ***логический (булевский) тип*** (*boolean*). Данные этого типа могут принимать только два значения: истина (*true*) или ложь (*false*).

Описание: <идентификатор>: boolean;

Пример: *var* x, y, flag: boolean;

Операции: *and, or, xor, not.* Значение логических операций для *x* и *y* представлены в порядке убывания приоритета выполнения [3] в таблице 3.

Таблица 3 – Выполнение логических операций

Логические переменные		*Логические операции*			
x	***y***	*not x*	*x and y*	*x or y*	*x xor y*
false	false	true	false	false	false
false	true	true	false	true	true
true	false	false	false	true	true
true	true	false	true	true	false

Стандартные функции для работы с логическими данными описаны в приложении 3.

Для хранения значения переменной логического типа потребуется 1 бит.

К простым типам данных относят типы, которые может определить сам пользователь (см. п. 2.5).

2.5 Пользовательские типы данных

Языки программирования характеризуются набором стандартных типов данных. Только язык Паскаль позволяет образовывать новые типы данных. Создание новых скалярных типов данных возможно двумя способами: ограничением преде-

лов изменения какого-нибудь уже известного скалярного типа данных (*интервальный* тип) или перечислением имен всех констант, принадлежащих новому скалярному типу данных (*перечисляемый* тип).

Новые типы данных, определенные пользователем, имеют некоторые общие свойства:

– для размещения данных этих типов необходимо по одному байту памяти, поэтому наборы значений этих типов не могут содержать более 256 элементов;

– запрещено присваивание переменным образованных типов данных значений из описания другого типа;

– к достоинствам новых типов данных следует отнести улучшение наглядности программы и экономию памяти.

Перечисляемый (перечислимый) тип задается перечислением всех значений, которые может принимать переменная этого типа. Значения перечисляются через запятую, а список значений заключается в круглые скобки.

Описание типа: type < имя типа > = (w_0, w_1, ..., w_n);

Описание переменных: var <идентификатор>: <имя типа>;

Возможно совмещенное описание:

var fakultet: (finans, agronomy, mehanich, agrochemistry);

Требования к использованию:

– значения w_0, w_1, ..., w_n должны быть константами-идентификаторами;

– значения w_0, w_1, ..., w_n должны быть различны;

– в списке идентификаторов первому из них присваивается порядковый номер – 0, второму –1 и т. д. Например, *ord* $(w_n) = n$;

– описание типа упорядочивает константы, а именно считается, что $w_i < w_k$ при $i < k$;

– запрещены операции ввода / вывода значений;

– к перечислимому типу применимы операции отношения: =, <>, <=, >=, <, >, а арифметические операции запрещены.

Пример: пусть определены новые типы данных:

type figura = (romb, kvadrat, krug, oval, treug);

var vid: fakultet; a1,a2: figura;

Переменные *a*1 и *a*2 могут принимать значения типа *figura*. Если переменной *a*1 присвоить значение *romb*, а переменной *a*2 значение *krug*, то результат выполнения операции сравнения *a1<a2* имеет истинное значение.

Стандартные функции: ord, pred, succ; процедуры *inc* и *dec* описаны в приложении 3.

Интервальный (диапазонный, ограниченный) тип определяется начальным и конечным значениями (границами), задающими диапазон значений этого типа.

Описание типа задается верхней и нижней границами диапазона, разделенными двумя точками:

type < имя типа > = <мин. значение>..<макс. значение>;

Описание переменных var <идентификатор>: <имя типа>; или

var <идентификатор>: <мин. значение>..<макс. значение>;

Скалярный тип, на который накладывается ограничение, называется *базовым* типом.

Требования к использованию:

– в качестве границ используются значения целого, символьного и строкового типов (кроме вещественного типа);

– значения границ должны быть константами, различными по значению и принадлежащими одному и тому же базовому типу;

– задание типа возможно, если значение нижней границы не превышает значения верхней границы;

– рекомендуется программно контролировать принимаемые значения переменных интервального типа через задание директивы компилятору, т. е. состояние {*$R+*};

– отключение состояния {*$R-*} уместно использовать в ситуации, когда проверки значений переменных интервального типа уже не нужны. Это позволит сократить количество проверок и тем самым ускорить выполнение программы.

Примеры определения интервальных типов данных и описания переменных интервального типа:

type bukva = ‘a’..‘я’;

```
    cifra = '0'..'9';
    interval = -10..50;
    day = 1..31;
    month = 1..12;
var index1, index2: interval;
  1_symbol, 2_symbol: bukva;
```

Описания переменных интервального типа можно упростить, совмещая описание переменных с их типом:

```
var index1, index 2: -10..50;
  1_symbol, 2_symbol: 'а'..'я';
```

Для интервального типа справедливо использование всех *стандартных функций*, применимых к соответствующему базовому типу (приложение 3).

2.6 Понятие выражения

Выражение представляет собой формулу, состоящую из значений констант, переменных, операций и функций, служащую для вычисления некоторого нового значения..

Операции и типы объектов данных, используемые в выражении, формируют *тип значения выражения*. Выражения бывают *арифметические* и *логические.*

В состав арифметических выражений могут входить следующие объекты данных:

- числовые значения;
- данные числовых типов;
- арифметические операции: +, –, *, /, *div*, *mod*;
- данные структурированных типов;
- арифметические функции;
- скобки.

Знаки операций: + (сложение), – (вычитание), * (умножение), / (деление) имеют обычный математический смысл. Операция *div* обозначает целочисленное деление (с отбрасыванием остатка), а операция *mod* обозначает взятие остатка от целочисленного деления. Знаки операций *div* и *mod* должны

всегда отделяться пробелами. Вычисление значения арифметического выражения осуществляется с учетом скобок и приоритета (по убыванию): *, /, *div, mod*, +, –.

Известно, что тип функции определяется типом результата ее работы. Обращение к функции имеет вид: *F(x),*

где *F* – имя функции, *x* – аргумент.

В программе можно использовать только значения стандартных функций (приложение 3), а нестандартные функции необходимо выразить через стандартные (таблица 4).

Примеры правильных арифметических выражений:

(*x* + 1) *div y*;

PI * *sqr* (*R*);

sqrt (*abs* (1 – *x*)).

Примеры неправильных арифметических выражений:

+ x – y – начинается со знака операции;

*x – * y* – содержит два знака операции подряд;

xdiv y – нет пробела перед операцией *div*;

cos (x) mod y – нецелый аргумент у операции mod;

(x + y) z – пропущен знак операции;

*z * (x – y* – пропущена закрывающая скобка.

Результат вычислений логических выражений может быть *true*, либо *false*. Логические выражения могут содержать следующие объекты данных:

- числовые значения;
- данные числовых, символьных и логических типов;
- арифметические операции: +, -, *, /, *div*, *mod*;
- логические операции: *and, or, not, xor;*
- операции отношения (сравнения): <, >, >=, <=, <>, =;
- данные структурированных типов;
- арифметические и логические функции;
- скобки.

Сравнивать можно только однотипные данные.

Примеры правильных логических выражений:

not (eof (f));

(otvet = ’Y’) or (otvet = ‘N’);

odd (y) or (y>=10);
(d.pol = ‘m’) and (d.voen = true).

Ошибки, допускаемые при построении логических выражений, почти такие же, как и в арифметических выражениях.

Таблица 4 – Преобразование нестандартной в стандартные функции

Нестандартная функция	*Стандартные функции*
a^x	$e^{x \ln a}, a > 0$
$\sqrt[n]{x}$	$e^{\frac{\ln x}{n}}$
$\log_a x$	$\frac{\ln x}{\ln a}$
$tg\ x$	$\frac{\sin x}{\cos x}$
$ctg\ x$	$\frac{\cos x}{\sin x}$
$arcsin\ x$	$arctg\ \frac{x}{\sqrt{1-x^2}}, 0 < x \le 1$
$arccos\ x$	$\frac{\pi}{2} - arctg\ \frac{x}{\sqrt{1-x^2}}, 0 < x \le 1$
$arcctg\ x$	$\frac{\pi}{2} - arctg\ x$

2.7 Структура программы

Вид программы на языке Паскаль определяет ее структуру. Программа записывается в виде последовательных строк и состоит из набора команд (*тело программы*), которые в программировании принято называть ***операторами***, а параметры команды называют ***операндами***.

Общий вид программы можно представить следующим образом:

[Заголовок программы]
[Раздел описаний]
begin
<Раздел операторов>
end.

Заголовок программы не является обязательным, но желательным элементом программы. Обычно состоит из зарезервированного слова *program* и имени программы. Может быть указан список параметров в круглых скобках и заканчивается обязательно точкой с запятой.

Параметрами программы обычно являются имена стандартных файлов ввода/вывода *Input* и *Output*, которые указывать не обязательно, так как действует принцип «по умолчанию»[5].

Если в качестве входных и выходных данных программы используются нестандартные файлы, то имена этих файлов могут быть указаны в качестве параметров всей программы.

[*Program* <имя программы> [(*Input, Output*)]];

где <имя программы> создается по правилам образования идентификаторов (см. п. 2.2).

Раздел описаний может отсутствовать, если в программе нет описаний объектов. Порядок размещения разделов описаний произвольный, возможно задание одинаковых разделов описаний. Например, *var* x: real; *var* y: char;

Описание типа должно предшествовать описанию переменной. Раздел описаний программы содержит следующие структурные единицы:

- ***определение констант*** начинается с зарезервированного слова *const* и служит для присваивания постоянных зна-

[5] *По умолчанию* – это понятие для обозначения значений параметров программы, которые установил разработчик. У пользователя есть возможность изменять эти параметры.

чений идентификаторам.
Пример: *const* dlina = 10.01;
GOST = ‘19.002-80 – схемы алгоритмов и программ';

- ***определение меток*** предназначено для перечисления через запятую меток, начинается со слова *label*. Метками могут быть целые числа без знака или идентификаторы и ставятся в программе перед оператором и отделяются двоеточием.

Пример: *label* 3, m1, m2, blok5;

- ***определение типов*** начинается с зарезервированного слова *type* и используется для объявления скалярных пользовательских и структурных типов данных.

Пример: *type* year = 1900..2000;
A = *array* [1..10] *of* real;

- ***определение переменных*** начинается с зарезервированного слова *var*, за которым перечисляются через запятую имена переменных, затем через двоеточие их тип.

Пример: *var* g: year;
t: A; i, j: integer;

- ***определение процедур и функций*** может начинаться с зарезервированных слов *procedure* или *function*. Содержит описание всех пользовательских подпрограмм, а стандартные подпрограммы не требуют предварительного описания.

Раздел операторов описывает действия, которые выполняются в процессе решения задачи, и обрамляется *операторными скобками begin* и *end* с точкой. Внутри скобок перечисляются через точку с запятой (*разделитель*) все операторы программы. Исключением является зарезервированное слово *end*, перед которым разделитель обычно не ставится.

Является обязательным разделом и, как правило, состоит из следующих частей:

- ввод данных (может отсутствовать) (см. п. 2.8);
- обработка данных (см. п. 2.10 - 2.15);
- вывод результатов обработки (см. п. 2.9).

Все операторы языка Паскаль можно разделить на три группы:

- ***операторы ввода/вывода данных***
  - ввод данных;
  - вывод данных;
- ***простые операторы***
  - оператор присваивания;
  - оператор безусловного перехода;
  - пустой оператор;
  - оператор вызова процедуры;
- ***структурные операторы***
  - составной оператор;
  - условный оператор;
  - оператор выбора;
  - цикл с параметром;
  - цикл с предусловием;
  - цикл с постусловием;
  - оператор присоединения.

2.8 Ввод данных

Программа получает входные данные «по умолчанию» из стандартного файла *Input*, который связан с клавиатурой.

Чтение данных осуществляет ***оператор ввода данных*** с клавиатуры.

read ([имя файла ввода], < список идентификаторов >);

Требования к выполнению:

- именем файла ввода «по умолчанию» считается стандартный файл ввода *Input* и обычно не указывается, иначе указывается имя файла ввода, откуда будут вводиться данные при выполнении программы;
- <список идентификаторов> – список имен переменных, перечисленных через запятую;
- существуют два равнозначных варианта задания операторов ввода данных: *read (x); read (y);* или *read (x, y);*

– разрешено вводить значения данных целого, вещественного, символьного и диапазонных типов;

– запрещено вводить данные логического и перечисляемого типов данных;

– вводимые значения переменных должны соответствовать описанному типу и количеству переменных;

– существуют два равнозначных варианта ввода данных с клавиатуры: через <*Пробел*>, а затем нажимается клавиша <*Enter*> или каждое значение через нажатие клавиши <*Enter*>, при этом курсор клавиатуры остается в той же строке после последнего значения.

Выполнение: программа приостанавливается при выполнении оператора ввода и на экран выводится курсор. ПК переходит в режим ожидания набора данных для переменных, имена которых указаны в списке ввода.

Если файл ввода уже заполнен вводимыми значениями, оператор ввода начинает чтение из этого файла и присваивание их именам переменных из списка ввода.

Пример использования:

var x: integer;

y: real; z: char;

Ввод значений переменных *x, y, z* может осуществляться двумя способами. Например:

– <10> <пробел> <-2.5> <пробел> <’*’> <*Enter*>;

– <-8> <*Enter*> <0.1> <*Enter*> <’*R*’> <*Enter*>.

Если после ввода данного из списка ввода надо перевести курсор клавиатуры на следующую строку, то используют оператор ввода *readln*.

2.9 Вывод данных

Вывод сообщений, промежуточных и конечных результатов работы программы осуществляет ***оператор вывода данных***.

write ([имя файла вывода], <список вывода>);

Требования к выполнению:

– именем файла вывода «по умолчанию» является стандартный файл вывода *Output*, иначе указывается имя файла, куда будут записаны данные результатов выполнения программы;

– списком вывода может быть любое количество данных: переменных, констант и выражений, перечисленных через запятую или сообщение, обрамленное в апострофы;

– вывод значения выражения последует после его вычисления;

– разрешено выводить значения данных целого, вещественного, символьного и диапазонных типов;

– запрещено выводить данные логического и перечисляемого типов данных;

– выводимые значения должны соответствовать описанному типу переменных;

– данные можно выводить с указанием формата. Для целочисленных типов данных:

write (<идентификатор>: <ширина поля>);

для вещественных данных:

write (<идентификатор>: <ширина поля 1>:< ширина поля 2>);

где <ширина поля 1> задает общее количество позиций для вывода значения идентификатора, учитывая точку;

<ширина поля 2> задает количество позиций, выделяемых под дробную часть вещественных данных.

Выполнение: выводит на экран монитора данные из списка вывода.

Для вывода с переводом курсора клавиатуры на следующую строку используют оператор вывода *writeln*.

Если у данных вещественного типа не указана <ширина поля 2>, то они будут выводиться в *научном формате*. Например, число 0.000123 представляется как 1.23E-04.

Вывод строк может осуществляться в *стандартном формате*, при этом выделяется количество позиций вывода, рав-

ное количеству символов, не считая апострофов, или с заданием ширины поля вывода, как и для целочисленных данных.
Примеры использования:
write ('Введите число'); – вывод сообщения перед оператором ввода;
writeln (‘y= ‘, sqrt (x – 1): 5: 2); – вывод комментария и значения арифметического выражения вещественного типа; с заданием ширины полей вывода и с переводом строки.

2.10 Программирование линейных процессов обработки информации

Операторы, которые в своей структуре не объединяют другие операторы, называются *простыми* операторами. К ним относятся операторы: пустой, присваивания, обращения к процедуре и оператор безусловного перехода[6]. Простые операторы используются для программирования линейных процессов обработки информации.

Оператор присваивания изменяет значение переменной из левой части на результат вычисления выражения из правой части.

<идентификатор>**:=** <выражение>;

Требования к выполнению:

– идентификатор из левой части и значение выражения из правой части должны быть одного и того же типа; исключением является присваивание вещественной переменной целочисленного значения;

– присваивать можно значения любых типов данных (кроме, файловых);

– значения переменных, связанных в выражение, должны быть определены до выполнения оператора присваивания.

[6] Оператор безусловного перехода не используется для программирования линейных процессов обработки информации, поэтому он будет рассмотрен в описании структурных операторов.

Выполнение: вычисляет значение выражения из правой части, если надо, и присваивает переменной из левой части.
Примеры использования:
*S := pi * sqr (R) * h;* – значение переменной *R* возводится в квадрат и умножается на значения констант *pi* и *h*. Результат присваивается переменной *S*;
f: = true; – логической переменной *f* присваивается *true*.

Задача 1. *Заданы значения x и y. Напишите программу вычисления значения z, используя стандартные функции, если*

$$z = \sqrt{\frac{3^x \sqrt[5]{|y|}}{\ln|y+1|}}$$

Программа решения задачи:

```
program pr_1;
var x, y,                                  {x, y – заданные числа}
a,                                         {a  – рабочая переменная}
z: real;                                   {z – результат}
begin
    write (‘Введите числа x и y’);
    readln (x, y);
    a:= exp (x * ln (3)) * exp (ln (abs (y + 1)) / 5) ;
    z:= sqrt (a / ln (abs (y)));
    write ('z= ', z : 6 : 2)
end.
```

Пояснения к программе:

Нестандартные функции: 3^x и $\sqrt[5]{|y|}$ выражены через стандартные функции *exp (x * ln (3));* и *exp (ln (abs (y)) / 5)* соответственно (см. таблицу 4).

С вводом *рабочей переменной* (*a*) упростилась запись выражения. Этот прием часто используется в записи сложных выражений.

Пустой оператор не выполняет никаких действий в программе и не обозначается зарезервированными словами.

Требования к выполнению:

– в качестве пустого оператора используется точка с запятой;

– оператор может располагаться в любой части программы согласно правилам синтаксиса языка;

– к пустому оператору может быть установлена метка.

Выполнение: служит для организации выхода из середины программы.

Оператор обращения к процедуре осуществляет вызов вспомогательной программы и передает список параметров, если он определен.

<имя процедуры> [(список параметров)];

Требования к выполнению:

– (список параметров) может отсутствовать;

– количество передаваемых параметров должно соответствовать количеству и типу параметров, описанных в процедуре.

Пример использования:

```
program prog;
var a, b: real;
procedure pr;
      begin
         write (‘********** ’);
      end;
  procedure summa (x, y);
  var x, y, z: real;
      begin
         z:= x + y;
         writeln (‘Сумма чисел равна ’, z : 6 : 2)
      end;
  begin
      pr;
      summa (a, b)
  end.
```

Задания для самостоятельного выполнения

1. Найдите абсолютную часть, вычислите квадратный корень, выделите целую и дробную части; и округлите до целых значение функции y(x) при заданном значении x.

№	y(x)	№	y(x)
1	$ln\sqrt[3]{\lvert x-1\rvert}-e^{x+1}$	**14**	$ln\sqrt[5]{\lvert x-e^{-(x+3)}\rvert}$
2	$log_2\lvert 1-3x^3\rvert+\sqrt{\lvert x\rvert}$	**15**	$\sqrt{lg\sqrt{\lvert 1-2x\rvert}}$
3	$\sqrt{ln\lvert x\rvert}+\sqrt[3]{\lvert x\rvert}$	**16**	$\sqrt{3{,}8^{2x-3}\,ln\lvert 1-x\rvert}$
4	$2{,}2^{x-1}-\sqrt[3]{ln\lvert x+1\rvert}$	**17**	$e^{x-3}-\sqrt{lg\ \lvert x-1\rvert}$
5	$e^{x-1}\,ln\sqrt{\lvert x-1\rvert}+3{,}2^{2x}$	**18**	$\sqrt{\lvert x-e^x\rvert+5{,}3^{x-1}}$
6	$\sqrt{2{,}1^x\,ln\lvert 1-x\rvert}$	**19**	$\sqrt{ln\lvert x\rvert+4{,}3^{x-1}}$
7	$e^x+\sqrt[5]{ln\lvert x-1\rvert}$	**20**	$5{,}2^{3x-2}-ln\sqrt{\lvert 2x-1\rvert}$
8	$3{,}5^{x+1}\,ln\sqrt{\lvert 1-x\rvert}$	**21**	$\sqrt{ln\lvert 2x-1\rvert+\sqrt[3]{\lvert 1-2x\rvert}}$
9	$ln\sqrt{\lvert x\rvert}-\sqrt[5]{\lvert x\rvert}$	**22**	$\sqrt{\lvert x\rvert\,ln\sqrt[5]{\lvert 1-x\rvert}}$
10	$e^{x+1}+\sqrt{log_2\lvert x+1\rvert}$	**23**	$ln\sqrt{\lvert x+\sqrt[3]{2\lvert x\rvert}\rvert}$
11	$\sqrt{ln\lvert x\rvert+2{,}4^{x+1}}$	**24**	$\sqrt{2{,}9^{x+1}\lvert 1-ln\lvert x\rvert\rvert}$
12	$\sqrt{\lvert x\rvert+e^x}-4{,}1^{3x}$	**25**	$\sqrt[3]{\lvert 1-x\rvert}\,ln\sqrt{\lvert x-1\rvert}$
13	$\sqrt{ln\lvert x-1\rvert+3{,}7^{x+1}}$	**26**	$\sqrt{3{,}1^{x-1}\lvert 1-e^{x-1}\rvert}$

2. Напишите программу вычисления значения выражения z при заданных значениях x и y.

№	z	№	z	№	z
1	$\frac{\log_3\lvert 1+x\rvert}{\ln(\lvert y\rvert+1)}$	**10**	$\frac{y-1}{\sqrt[3]{1+x^2}}$	**19**	$\frac{2^x-1}{2y^2+1}$
2	$\frac{\sqrt[3]{\lvert y^3\rvert+2}}{x+y}$	**11**	$\frac{x^2-5x+6}{lg\lvert y+1\rvert}$	**20**	$\frac{y-1}{\log_2(x^2+1)}$
3	$\frac{\sqrt{\lvert y\rvert+1}}{3^x}$	**12**	$\frac{y-x}{\sqrt[5]{y^2+1}}$	**21**	$\frac{\sqrt[3]{\lvert y-1\rvert}}{1-ln\lvert x\rvert}$
4	$\frac{\log_7\lvert x^3\rvert}{y}$	**13**	$\frac{\log_2(\lvert x\rvert+2)}{y-1}$	**22**	$\frac{2^y-1}{2x^3+1}$
5	$\frac{\sqrt{\lvert x\rvert}+3}{lg(y+1)}$	**14**	$\frac{2^y-1}{x^2+2}$	**23**	$\frac{\sqrt[3]{y^2+3}}{1+2x}$
6	$\frac{ln\lvert y+x\rvert}{\sqrt[3]{2\lvert x\rvert+1}}$	**15**	$\frac{ln\lvert x\rvert+1}{lg(y+1)}$	**24**	$\frac{\sqrt[3]{\lvert x\rvert+1}}{3-2ln\lvert y\rvert}$
7	$\frac{x+y}{lg(\sqrt{\lvert y\rvert}+1)}$	**16**	$\frac{x-ln\lvert y\rvert}{\sqrt[3]{\lvert y+1\rvert}}$	**25**	$\frac{ln\lvert x\rvert-y}{\log_5(3\lvert x\rvert+1)}$
8	$\frac{2x+1}{lg(y^2+2)}$	**17**	$\frac{x+1}{lg\sqrt{\lvert y+1\rvert}}$	**26**	$\frac{\sqrt{\lvert y\rvert}}{1+\log_2\lvert x+1\rvert}$
9	$\frac{\sqrt[5]{\lvert x\rvert+5}}{x^3+y}$	**18**	$\frac{lg(1+\lvert y\rvert)}{\sqrt{\lvert x-1\rvert}}$		

Контрольное задание 1

Даны x и y. Напишите программу вычисления значения выражения z, добавив функцию вычисления модуля, где это необходимо.

№	z	№	z
1	$\frac{\sqrt{arctg\ x}}{lg\ x-\sqrt{y}}$ *при x = 1, y = 1*	**14**	$\frac{\sqrt[3]{y}+tg\ x}{\sqrt{e^{x+y}}}$ *при* $x=\frac{\pi}{3}$, *y = 5*
2	$\frac{\sqrt{x-\log_2 y}}{e^{y-1}-\sqrt{xy}}$ *при x = 4, y = 5*	**15**	$\frac{(1+y)^2+e^y}{Cos\ x+\lvert Ctg\ x\rvert}$ *при* $x=\frac{5\pi}{3}$, *y = 2*
3	$\sqrt{\frac{e^x+1}{ctg\ x}}-y\log_x y$ *при* $x=\frac{\pi}{4}$, *y = 1*	**16**	$\frac{\sqrt{y-1}-\sqrt[3]{\lvert y\rvert}}{\ln x^2+y}$ *при x = 3, y = 4*
4	$\frac{\sqrt{1+e^y}}{lg\ tg\ x}$ *при x = 1, y = 4*	**17**	$\frac{lg\sqrt{1+y^3}}{arcctgx}$ *при x = -1, y = 2*
5	$\frac{xy-\lvert x+\sqrt{y}\rvert}{lg\left(y^2-1\right)}$ *при x = 1, y = 3*	**18**	$\frac{arccos\ x-\ln y^2}{\sqrt{y-1}}$ *при x = 0, y = 3*
6	$\log_3\frac{e^{x+1}}{\sqrt{y}+x^2}$ *при x = 2, y = 1*	**19**	$\frac{Cos\ x}{lg\ y\sqrt{1+Sin\ x}}$ *при* $x=\frac{\pi}{6}$, *y = 8*

7	$\frac{\lvert x\rvert+\sqrt{y+1}}{\sqrt{1-y\lg x}}$ *при x = 1, y = 2*	**20**	$\frac{x\arcsin\sqrt{x}}{e^{y+1}\sqrt{y-1}}$ *при x = 0.5, y = 2*
8	$\lg\sqrt{\frac{x+e^{2y}}{x^3}}$ *при x = 5, y = 2*	**21**	$\frac{\sqrt{e^{y+2}+x}+2arcctg\ y}{Cos\ x}$ *при x = 1, y = 12*
9	$\frac{\sqrt{x}+e^x}{\lg^2 y}$ *при x = 2, y = 5*	**22**	$\frac{e^{\sqrt{y+1}}\sqrt[3]{y+x}}{ln(x+y)}$ *при x = 11, y = 9*
10	$\frac{e^x+ln\ y-\sqrt{x-1}}{arccos\ y}$ *при x = 2, y = 0.5*	**23**	$\frac{e^{-x}-\log_5(y-1)}{\sqrt{e^{x-1}-1}}$ *при x = 3, y = 3*
11	$\frac{\sqrt{e^x+1}}{\sqrt{y-1}+arccos\ x}$ *при x = 0, y = 2*	**24**	$arctg\ x+\frac{\sqrt{e^{2y}-1}}{arcsin\ x}$ *при x = 0.5, y = 1*
12	$\frac{\lvert 3x^2-y\rvert}{\lg x+e^{x-1}};$ *при x = 9, y = 7*	**25**	$\frac{lg(x+1)}{\sqrt{\lvert 1-e^y\rvert}}$ *при x = 5, y = 5*
13	$\frac{\sqrt{e^{3x}+e^{2y}}}{arcsin\ x}$ *при x = 0.5, y = 1*	**26**	$\frac{ln\ y+e^x}{\sqrt{1+2^{xy}}}$ *при x = 2, y = 2*

2.11 Программирование разветвляющихся процессов обработки информации

Программирование разветвляющихся процессов обработки информации реализуется через использование структурного[7] условного и составного операторов; простого оператора безусловного перехода.

Составной оператор за счет задания порядка выполнения операторов внутри скобок делает программу более наглядной.

begin <оператор 1>; …; <оператор *N*> ***end***;

Требования к выполнению:

– выполнение следующего оператора должно начаться сразу же после того, как закончится выполнение предыдущего;

– *end* рассматривается как разделитель (точка с запятой), поэтому перед *end* точка с запятой обычно не ставится.

Выполнение: составной оператор объединяет операторы в блок.

Условный оператор позволяет реализовать разветвление в зависимости от результата проверки условия.

Описание может быть задано одной из двух форм:

– *полная:* ***if*** <условие> ***then*** <оператор 1> ***else*** <оператор 2>; – *краткая:* ***if*** <условие> ***then*** <оператор 1>;

Требования к выполнению:

– <условие> является выражением логического типа;

– сложные условия могут задаваться с помощью операций отношения (<, >, <>, <=, >=) и соединяться между собой логическими операциями (*not, and, or, xor*);

– <условие> всегда проверяется на истинность, даже если в логическом условии указывается только имя переменной

[7] Каждый из структурных операторов представляет собой сложную структуру из других операторов.

логического типа;

– в качестве <оператор 1> и <оператор 2> используется один оператор. Если количество операторов больше чем один, то используется составной оператор.

Выполнение: условный оператор вычисляет результат <условие>, если это значение есть истина (*true*), то выполняется <оператор 1>, и на этом выполнение заканчивается; иначе выполняется <оператор 2>, и выполнение условного оператора заканчивается.

Во втором случае, если значение выражения <условие> есть истина (*true*), то выполняется <оператор 1>, иначе действие не производится, т. е. в качестве <оператор 2> взят пустой оператор.

Примеры использования:

if (*y* > 0) *and* (*x* <> 0) *then* z:= ln (y) * x;

if f *then* s:= s + x *else* s:= s – x;

Задача 2. *Выведите число, полученное выписыванием в обратном порядке цифр заданного целого трехзначного числа.*

Программа решения задачи:

```
program pr_2;
var x ,                                          {x – заданное число}
    x1, x2,                {x1 – число единиц, x2 – число десятков}
    x3: integer;                          {x3 – число сотен результата}
    y: char;                                          {y – знак числа}
 begin
    writeln (‘Введите трехзначное число’); readln (x);
    write (‘Заданное число ’, x, ‘ в обратном порядке равно ‘);
    if x < 0 then begin x:= abs (x); y:= ’-‘ end  else y:=’ ‘;
    x1:= trunc (x/100);
    x:= x – x1 * 100;
    x2:= trunc (x/10);
    x3:= x – x2 * 10;
    write (y, x3, x2, x1)
 end.
```

Пояснения к программе:

У заданного трехзначного числа целого типа следует учесть знак (*y*). Это и выясняется с использованием условного оператора.

Оператор безусловного перехода[8] передает управление к оператору, имеющему метку.

goto <метка>;

Требования к выполнению:

– <метка> должна быть обязательно описана в разделе описаний меток той программы, где она используется;

– область действия метки ограничена тем блоком, где она описана;

– переход может осуществляться только в пределах того блока, где метка описана;

– переход за пределы или внутрь другого блока вызывает программное прерывание.

Пример использования:

```
label metka;
 begin    if <условие>
          then goto metka;
          …
   metka: <оператор>;
 end.
```

Вместо использования оператора перехода для организации разветвляющихся процессов, можно применить конструкции вида: *if – then – else* или *case* (см. п. 2.12).

Оператор перехода иногда используется для организации повторений серии действий, т. е. когда переход по метке осуществляется к оператору, расположенный ближе к началу программы. В этом случае, можно заменить оператор перехода конструкциями: *while* или *repeat* (см. п. 2.14, п. 2.15).

[8] Применение оператора безусловного перехода обычно приводит к нарушению принципов структурного программирования и усложняет чтение и отладку программ.

Задания для самостоятельного выполнения

1. Задано число x. Сравните выражения и выясните: a > b (ответ в виде значения: true или false).

№	*a*	*b*
1	$4^x\sqrt[3]{\lvert x-1\rvert}-\sqrt{3\lvert x\rvert}$	$2^x\sqrt{\lvert x-1\rvert}+\sqrt{\lvert x+1\rvert}$
2	$3\sqrt{\lvert x+1\rvert}-\sqrt[3]{e^{x+1}}$	$\sqrt{\lvert x-1\rvert}+\sqrt{e^x}$
3	$lg\lvert x-1\rvert+\sqrt[3]{2^x}$	$lg\lvert 3x^2-1\rvert-\sqrt[3]{2^{x-1}}$
4	$\sqrt{\lvert 2x^3+3^{-x}\rvert}+x\sqrt{\lvert x\rvert}$	$\sqrt{\lvert 5x^5+3^x\rvert}-\sqrt{\lvert x\rvert}$
5	$5e^{x+1}-2lg\sqrt[3]{\lvert x\rvert}$	$2e^{x+1}+3log_2\sqrt[3]{\lvert x\rvert}$
6	$3^{2x}+lg\sqrt{1+e^{2x}}$	$4^{2x}-log_3\sqrt{1+e^{2x}}$
7	$5\sqrt{lg\ \lvert 1-e^x\rvert}-\sqrt[3]{2\lvert x\rvert}$	$\sqrt{lg\ \lvert 1-e^x\rvert}+\sqrt[3]{\lvert x\rvert}$
8	$3log_2\lvert 1-x\rvert+\sqrt{2^x}$	$5lg\lvert 1-x\rvert-\sqrt[5]{2^x}$
9	$4e^{x+1}-2\sqrt{log_3\lvert x-1\rvert}$	$2e^{x+1}+\sqrt{lg\lvert x-1\rvert}$
10	$2lg\sqrt{2^{2x}}+3\lvert 1-x\rvert$	$5lg\sqrt{2^{2x}}-2\lvert 1-x\rvert$
11	$3\sqrt[5]{\lvert x-1\rvert}+log_3\,2^{x-1}$	$3\sqrt{\lvert x+1\rvert}-log_5\,2^x$
12	$6\sqrt{lg\lvert x-\sqrt{\lvert x\rvert}\rvert}-2\sqrt[3]{\lvert x-1\rvert}$	$\sqrt{log_5\lvert x-\sqrt{\lvert x\rvert}\rvert}+\sqrt[3]{\lvert x-1\rvert}$

13	$ln\sqrt{\lvert 1-2x\rvert+\sqrt{3\lvert x\rvert}}$	$2\,ln\sqrt{\lvert 1-2x\rvert-\sqrt{\lvert x\rvert}}$
14	$3\sqrt{\lvert xe^{x+1}\rvert}-2\sqrt[5]{\lvert x-1\rvert}$	$2\sqrt{\lvert xe^{x}\rvert}+\sqrt[5]{\lvert x+1\rvert}$
15	$3^{x+1}+lg\left\lvert\sqrt{3\lvert x\rvert^{3}}\right\rvert$	$5^{x+1}-log_{5}\left\lvert\sqrt{2\lvert x\rvert}\right\rvert$
16	$5xe^{x-1}-2\sqrt{ln\lvert x+1\rvert}$	$3e^{x-1}+\sqrt{ln\lvert x-1\rvert}$
17	$2\,lg\ x^{4}+\sqrt[5]{\lvert x-1\rvert}$	$4\,lg\ 2^{x}-2\sqrt[3]{\lvert x-1\rvert}$
18	$5\sqrt{ln\lvert x\rvert+5^{x-1}}-2\sqrt[4]{\lvert x\rvert}$	$3\sqrt{ln\lvert x\rvert+3^{x-1}}+\sqrt[4]{\lvert x\rvert}$
19	$\sqrt{3\lvert x-1\rvert+2\sqrt[3]{e^{1-x}}}$	$\sqrt{4\lvert x-1\rvert-\sqrt[3]{e^{-x}}}$
20	$\sqrt{2\left\lvert 3x-e^{x-5}\right\rvert-2^{x-1}}$	$\sqrt{\left\lvert x-e^{x-1}\right\rvert+2^{x-1}}$
21	$3\sqrt[3]{\lvert x\rvert}+4lg\sqrt{\lvert x\rvert}$	$6\sqrt[3]{\lvert x\rvert}-3log_{3}\sqrt{\lvert x-1\rvert}$
22	$\sqrt{4\sqrt[5]{\lvert 1-x\rvert}-2^{3-x}}$	$\sqrt{2\sqrt[3]{\lvert 1-x\rvert}+2^{x+1}}$
23	$log_{2}\sqrt[3]{2^{x+1}+\lvert 1-3x\rvert}$	$log_{2}\sqrt{2^{x+1}-\lvert 1-2x\rvert}$
24	$6^{x-1}-\sqrt{\left\lvert x+\sqrt[3]{\lvert x\rvert}\right\rvert}$	$2^{x}+\sqrt{\left\lvert 2x+\sqrt{\lvert x+1\rvert}\right\rvert}$
25	$\sqrt{log_{4}\ ln\lvert x\rvert+e^{x}}+\sqrt[3]{\lvert x-1\rvert}$	$\sqrt{ln\,log_{3}\lvert x\rvert+2^{x}}-\sqrt{\lvert x\rvert}$
26	$\sqrt{lg\lvert x-1\rvert}-\sqrt[3]{\lvert x-1\rvert}$	$log_{\ 5}\sqrt{\left\lvert 1-2^{x}\right\rvert}+2\sqrt{\lvert x-1\rvert}$

2. Дано четырехзначное натуральное число N.

1 Вычислите сумму и разность первой и последней цифры числа;
2 Выясните, является ли сумма цифр четным числом?
3 Вычислите сумму цифр числа;
4 Переставьте цифры числа так, чтобы образовалось максимальное число;
5 Вычислите произведение нечетных цифр числа;
6 Выясните, все ли цифры числа различны?
7 Вычислите произведение цифр числа;
8 Переставьте цифры числа так, чтобы получилось минимальное число;
9 Найдите целую и дробную части частного первой и последней цифр числа;
10 Выясните, образуют ли цифры числа арифметическую прогрессию?
11 Вычислите сумму четных цифр числа и произведение нечетных;
12 Найдите наибольшую цифру в записи числа и поставьте ее на последнее место;
13 Выясните, образуют ли цифры числа геометрическую прогрессию?
14 Найдите разность и количество цифр числа, кратных 5?
15 Найдите наименьшую цифру в записи числа и поставьте ее на первое место;
16 Вычислите сумму натуральных логарифмов цифр числа;
17 Упорядочите цифры числа по возрастанию значений;
18 Выясните, каких цифр числа больше четных или нечетных?
19 Вычислите целую и дробную части частного второй и третьей цифры числа;
20 Выясните, все ли цифры числа четные?
21 Вычислите произведение квадратных корней цифр числа;

22 Найдите сумму и количество цифр числа, кратных 3?
23 Выясните, равна ли сумма двух первых цифр сумме двух его последних цифр?
24 Упорядочите цифры числа по убыванию значений;
25 Вычислите произведение десятичных логарифмов цифр числа;
26 Выясните, читается ли число одинаково слева направо и справа налево?

3.Заданы координаты N точек плоскости, не лежащих на осях координат:

а) выясните, найдется ли хоть одна сторона фигуры, параллельная одной из осей координат? Если да, то вычислите ее длину; б) определите:

0 при $N = 4$, принадлежит ли трапеция третьей или четвертой координатным четвертям?
1 при $N = 3$, принадлежит ли треугольник второй или третьей координатным четвертям?
2 при $N = 5$, принадлежит ли пятиугольник первой или четвертой координатным четвертям?
3 при $N = 4$, принадлежит ли прямоугольник третьей или четвертой координатным четвертям?
4 при $N = 3$, принадлежит ли прямоугольный треугольник первой или четвертой координатным четвертям?
5 при $N = 4$, принадлежит ли квадрат первой или второй координатным четвертям?
6 при $N = 3$, принадлежит ли треугольник второй или третьей координатным четвертям?
7 при $N = 4$, принадлежит ли четырехугольник третьей или четвертой координатным четвертям?
8 при $N = 5$, принадлежит ли пятиугольник первой или четвертой координатным четвертям?
9 при $N = 4$, принадлежит ли параллелограмм первой или второй координатным четвертям?

Решите задачу при заданных координатах точек $x_1, y_1 \ldots x_5, y_5$ *по вариантам:*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
x_1	-4,2	-3,2	6,1	-1,6	2,3	-3,1	-2,6	-2,1	4,5	-1,3
y_1	-1,1	4,7	3,2	-4,3	-1,5	1,2	3,0	-3,4	-0,8	4,5
x_2	-8,3	-7,4	8,2	-1,6	9,4	-3,1	-2,6	4,3	8,2	5,7
y_2	-4,2	-1,3	3,8	-2,2	-1,5	5,2	-2,0	-2,2	-2,9	4,5
x_3	5,5	-3,2	8,2	7,1	2,3	0,9	-7,1	8,1	4,5	7,7
y_3	-4,2	-2,1	1,9	-2,2	4,8	1,2	6,2	-8,7	3,2	2,5
x_4	3,3		6,5	7,1		0,9		-1,2	7,3	0,7
y_4	-1,1		-1,2	-4,3		5,2		-7,6	4,1	2,5
x_5			4,3						9,4	
y_5			1,7						1,2	

2.12 Программирование ветвящихся процессов обработки информации

Процессы обработки данных, где осуществляется выбор одной из нескольких (больше двух) ветвей, называются *множественным ветвлением*. Программирование процессов множественного ветвления осуществляется с помощью оператора выбора.

Оператор выбора позволяет перейти к <оператору> в зависимости от значения < выражения>.

```
case <выражение> of
                    <метка 1>: <оператор 1>;
                    <метка 2>: <оператор 2>;
                    <метка 3>: <оператор 3>;
                                …
                    <метка N>: <оператор N>;
                       else <оператор N+1>
end;
```

Требования к выполнению:

– метки являются константами одного и того же скалярного типа, причем разного значения;

– типы данных <выражение> и <метка 1>, …, <метка *N*> должны совпадать;

– если значения меток различны, а <операторы> совпадают, то можно перечислить данные метки через запятую;

– <оператор> выполняет одно действие. Если выполняется серия действий, то используется составной оператор.

Выполнение: вычисляет, если нужно, значение выражения. Если значение совпадает с *i*-той меткой, то выполняется *i*-тый оператор, иначе выполняется <оператор *N*+1>.

Пример использования: var f: boolean;

```
begin f:=true;
      case f of
              true: write (' true');
              false: write (' false')
      end;
```

Задача 3. *Для натурального числа k (k ≤ 50) напечатайте фразу: «На рейс зарегистрировали k пассажир(?)», согласовав окончание слова после значения k.*

Программа решения задачи:

```
program pr_3;
var k: 1..50;                          {k – количество пассажиров}
 begin
      {$R+}
      write ('Введите количество пассажиров '); readln (k);
      write (' На рейс зарегистрировали ', k);
      case k of
              1, 21, 31, 41: write (' пассажир');
              2..4, 22..24, 32..34, 42..44: write (' пассажира');
              5..20, 25..30, 35..40, 45..50: write (' пассажиров')
      end
      {$R-}
 end.
```

Пояснения к программе:

В программе использован способ программного задания директивы компилятору: включение директивы {*$R+*} проверки принадлежности значения переменной (*k*) заданному диапазону значений [1..50]. Другой способ – через пункты главного меню *Turbo Pascal:* <*Options*> <*Compiler*> *[ ]* ***R****ange checking* <*OK*> (см. гл. 1 п. 1.3). В конце программы желательно отключать директиву компилятору: *{$R-}.*

Задача 4. *Задано целое число x* ($x \le 100$). *Вычислите значение выражения y с использованием условного оператора и оператора выбора.*

$$y = \begin{cases} \dfrac{x^2}{\sqrt{x-1}}, если \;\; x \in (1;5] \\ \cos x, \; если \;\; x = 0 \\ \log_3 x, если \;\; 5 < x \le 9 \\ 2x^2, иначе \end{cases}$$

Программы решения задачи:

с условным оператором	с оператором выбора

program pr4_1;
var y: real;
x: 1..100;
begin
{$R+}
write ('Введите x'); read (x);
if (x > 1) *and* (x <= 5) *then*
y:= sqr (x) / sqrt (x – 1);
else if x = 0 *then* y:= cos (x)
else if (x > 5) *and* (x <= 9)
then y:= ln (x) / ln (3)
else y:= 2 * sqr (x);
write ('y = ', y : 5 : 2)
{$R-}
end.
	program pr4_2;
	var y: real;
	x: 1..100;
	begin
	{$R+}
	write ('Введите x'); read (x);
	case x *of*
	2..5: y:= sqr (x) /sqrt (x – 1);
	0: y:= cos (x);
	6..9: y:= ln (x) / ln (3)
	else y:= 2 * sqr (x);
	end;
	write ('y = ', y : 5 : 2)
	{$R-}
	end.

Пояснения к программе:

Проверка всех условий в программе *pr*4_1 выполняется с использованием *вложенных условных операторов*. В этом случае неясно, к какому *then* относится *else*, поэтому принято соглашение: *else* относится к самому последнему *then.*

Использование вложенных условных операторов заметно усложняет структуру программы, но зато производится на одну проверку меньше методом исключения третьего.

Проверка условий в программе *pr*4_2 поручена оператору выбора, но переменная *x* не может быть вещественной.

Нестандартная функция $log_3(x)$ выражена через стандартную функцию *ln (x*) (см. таблицу 4).

Задания для самостоятельного выполнения

1. Для натуральных чисел m и n, где m, n =1..50 напечатайте фразу, согласовав окончание слов после значений m и n:

1 В аудитории *m* стул(?) и *n* стол(?);
2 На смотре самодеятельности *m* юнош(?) и *n* девуш(?);
3 В книге *m* рисун(?) и *n* схем(?);
4 На диске *m* килобайт(?) текста и *n* гигабайт(?) фото;
5 После сессии в группе *m* отличник(?) и *n* двоечник(?);
6 По математике изучили *m* аксиом(?) и *n* теорем(?);
7 В зимнюю сессию *m* экзамен(?) и *n* зачет(?);
8 На практику определили *m* бакалавр(?) и *n* магистр(?);
9 В тексте *m* абзац(?) и *n* слов(?);
10 По истории изучили *m* событи(?) и *n* факт(?);
11 В деканат вызвали *m* студент(?) и *n* старост(?);
12 На конференцию пришли *m* врач (?) и *n* ординатор(?);
13 В ДНД дежурили *m* студент(?) и *n* преподавател(?);
14 На встречу пришли *m* депутат(?) и *n* избирател(?);
15 В программе описывается *m* переменн(?) и *n* констант(?);

16 Студенты обучаются по *m* направлени(?) и *n* специальност(?);

17 На соревнованиях *m* спортсмен(?) и *n* юниор(?);

18 По программе изучается *m* метод(?) и *n* способ(?);

19 В программе описывается *m* массив(?) и *n* множеств(?);

20 На ученом совете *m* профессор(?) и *n* доцент(?);

21 В программе описывается *m* строк(?) и *n* файл(?);

22 На флешке *m* гигабайт(?) видео и *n* мегабайт(?) музыки;

23 По географии изучили *m* остров(?) и *n* полуостров(?);

24 В файле табличного процессора *m* лист(?) и *n* таблиц(?);

25 На курс зачислили *m* абитуриент(?) и *n* льготник(?);

26 В поход записали *m* турист(?) и *n* инструктор(?);

2. Задано значение переменной x:

1 являющейся натуральным числом ($x \leq 100$) и определяющей меры жидкостей (в литрах). Дайте для этого числа наименование. Например: 1 – литр, 5 литров, 10 – декалитр, 100 – 1 гектолитр и т. д.

2 являющейся символом. Определите, является ли заданный символ буквой английского алфавита, цифрой или специальным символом.

3 являющейся натуральным числом ($x \leq 1$ млн.) и определяющей меры веса (в граммах) с коэффициентом 10. Дайте для этого числа наименование: «грамм», «килограмм», «центнер», «тонна». Например: 1 грамм, 10 граммов, 1000 – 1 килограмм и т. д.

4 являющейся символом. Определите, является ли заданный символ прописной или строчной буквой английского алфавита, или цифрой.

5 являющейся натуральным числом ($x \leq 100$) и определяющей меры длины (в миллиметрах). Дайте для этого числа наименование: «миллиметр», «сантиметр», «дециметр». Например: 1 миллиметр, 3 миллиметра, 10 – 1 сантиметр и т. д.

6 являющейся буквой. Определите, является ли она первой буквой названия цвета в радуге, и выведите название цвета.

7 являющейся натуральным числом ($x \leq 1000$) и определяющей меры площади (в кв. м) с коэффициентом 10. Дайте для этого числа наименование. Например: 1 кв. метр, 10 кв. метров, 100 – 1 ар, 1000 – 1 гектар.

8 являющейся буквой. Определите, является ли заданный символ гласной или согласной буквой английского алфавита.

9 являющейся натуральным числом ($x \leq 2^{40}$) и определяющей меры объема данных (в байтах) с коэффициентом 2^5. Дайте для этого числа наименование: «байт», «килобайт», «мегабайт», «гигабайт», «терабайт». Например: 1 байт, 2^5 байт, 2^{10} – 1 килобайт, 2^{20} – 1 мегабайт и т. д.

10 являющейся буквой. Определите, является ли она первой буквой названия месяца, и выведите название времени года.

11 являющейся натуральным числом ($x \leq 100$) и определяющей меры времени (в годах). Дайте для этого числа наименование: «год», «лет», «десятилетие», «индикт», «век». Например: 3 года, 5 лет, 15 – 1 индикт и т. д.

12 ($0 \leq x < 24$), определяющее время суток (в часах). Дайте для этого числа наименование: «утро», «день», «вечер» и «ночь».

13 являющейся натуральным числом ($x \leq 100$) и определяющей меры измерения углов (в градусах). Дайте для этого числа наименование: «острый», «прямой», «тупой», «развернутый», «выпуклый», «невыпуклый», «полный». Например: 1..89 – острый, 90 – прямой и т. д.

14 ($1 \leq x \leq 12$), определяющей номер месяца. Определите, к какому кварталу года относится месяц: «1 квартал», «2 квартал», «3 квартал» или «4 квартал».

15 являющейся натуральным числом ($x \leq 100$) и определяющей меры жидкостей (в миллилитрах). Дайте для этого

числа наименование. Например: 2 миллилитра, 10 – 1 сантилитр, 100 – 1 децилитр и т. д.

16 ($0 \le x < 24$), определяющей время суток (в минутах). Дайте для этого числа наименование: «завтрак», «обед», «полдник» и «ужин».

17 являющейся натуральным числом ($x \le 100$) и определяющей меры времени (в сутках). Дайте для этого числа наименование: «день», «неделя», «декада», «месяц», «квартал». Например: 2 дня, 7 – 1 неделя, 28..31 – 1 месяц и т. д

18 ($1 \le x \le 12$), определяющей номер месяца. Определите, к какому времени года относится месяц: «зима», «весна», «лето» или «осень».

19 являющейся натуральным числом ($x \le 100$) и определяющей меры расстояния (в метрах). Дайте для этого числа наименование. Например: 4 метра, 10 – 1 декаметр, 100 – 1 гектометр и т. д.

20 ($0 < x \le 100$) и определяющей возраст в годах. Дайте для этого числа название периода жизни: «детство», «отрочество», «юность», «молодость», «зрелость» или «старость».

21 являющейся натуральным числом ($x \le 2^{40}$) и определяющей меры объема данных (в килобайтах) с коэффициентом 2^{10}. Дайте для этого числа наименование: «килобайт», «мегабайт», «гигабайт», «терабайт», «петабайт». Например: 1 килобайт, 2^{10} – 1 мегабайт и т. д.

22 ($0 < x \le 360$ и $0 < y \le 360$) и определяющие координаты точки в градусах. Дайте название четверти, где лежит точка: «1 четверть», «2 четверть», «3 четверть» или «4 четверть».

23 являющейся натуральным числом ($x \le 2^{40}$) и определяющей меры объема данных (в мегабайтах) с коэффициентом 2^{10}. Дайте для этого числа наименование: «мегабайт», «гигабайт», «терабайт», «петабайт», «эксабайт». Например: 1 мегабайт, 2^{10} – 1 гигабайт и т. д.

24 ($1 \le x \le 31$), определяющей номер дня. Определите, к какой декаде месяца относится день: «1 декада», «2 декада» или «3 декада».

25 являющейся натуральным числом ($x \le 100$) и определяющей фигурное число. Дайте для этого числа название. Например: 1, 2, 3, 5, 11.. «простые», 4, 6, 8… «плоские», 8, 12, 16… «телесные».

26 ($1 \le x \le 31$), определяющей номер лунного дня. Дайте названия дням по фазам луны в лунном календаре. Например: «1 - новолуние», «15 - полнолуние», «растущая луна», «убывающая луна» и т. д.

Контрольное задание 2

Задано значение целого числа x ($x \le 50$). Напишите программы вычисления значения выражения y с использованием условного оператора и оператора выбора.

1
$$y = \begin{cases} x^2 - 5x + 6, \text{ при } x \in [2;3] \\ \arccos x, \text{если } x = 1 \text{ или } x = -1 \\ (2-x)^3, \text{иначе} \end{cases}$$

2
$$y = \begin{cases} x^2 - x - 6, \text{при } x = -2 \text{ или } x = 3 \\ \arccos x, \text{если } x \in [-1;0] \\ lg|2x|, \text{иначе} \end{cases}$$

3
$$y = \begin{cases} arcctg\ x, \text{при } x \in [0;1] \\ x^2 - 2x - 3, \text{при } x = -1 \text{ или } x = 3 \\ \sqrt[3]{|1-x|}, \text{иначе} \end{cases}$$

4
$$y = \begin{cases} x^2 - 3x + 2, \text{при } x \in [1;2] \\ lg(2x-1), \text{если } 2 < x < 10 \\ 2\sqrt[5]{|x-1|}, \text{иначе} \end{cases}$$

5

$$y=\begin{cases}\sqrt{x(3x^2+1)},\ если\ \ x\in(1;5)или\ x\in(5;10]\\ x^2-6x+5,при\ x=5\ или\ x=1\\ 2^{3x},иначе\end{cases}$$

6

$$y=\begin{cases}x^2-4x-21,при\quad x=7\ или\ x=-3\\ \arccos x,при\ \ x\in[0;1]\\ \ln|x-1|,иначе\end{cases}$$

7

$$y=\begin{cases}x^2+3x-18,\ при\ x=3\ или\ x=-6\\ \cos x,если\ \ x=0\\ \sqrt[5]{|1-x|},\ иначе\end{cases}$$

8

$$y=\begin{cases}x^2-\ 6x+27,\ при\ x=-9\ или\ x=-3\\ arctg\ x,при\ x=-1\ или\ x=1\\ e^{2x},иначе\end{cases}$$

9

$$y=\begin{cases}x^2-7x+6,при\ x=6\ или\ \ x=1\\ \log_3(x-1),если\ 1<x\le 5\\ \ln|x-2|,иначе\end{cases}$$

10

$$y=\begin{cases}x^2-5x+4,при\ x=4\ или\ x=1\\ \sin 2x,при\ \ x=0\\ \sqrt[3]{|1-x|},иначе\end{cases}$$

11

$$y=\begin{cases}x^2+3x-4,при\ x=-4\ или\ x=1\\ \arcsin x,если\ x\in[0;1]\\ \ln(x^2-1),иначе\end{cases}$$

12

$$y=\begin{cases}x^2+8x+15,при\ x=-5\ или\ x=-3\\ \sqrt[3]{x+7},если\ \ x\in(-5;-3)или\ x\in[-2;10]\\ e^{x+1},иначе\end{cases}$$

13
$$y=\begin{cases} Sin\ x, если\ x\in[0;2] \\ x^2-4x-5, при\ x=5, x=-1; \\ lg\sqrt{|x-1|}, иначе \end{cases}$$

14
$$y=\begin{cases} x^2-7x+12,\ при\ x\in[3;4] \\ arccos\, x,\ при\ x=-1\ или\ x=1 \\ \sqrt{|x-1|},\ иначе \end{cases}$$

15
$$y=\begin{cases} x^2-2x-8, при\ x=-2, x=4 \\ \sqrt[3]{x-1}-3x, если\ \ x\in(0;4)\ или\ x\in(4;30] \\ \sqrt{1+e^x}, иначе \end{cases}$$

16
$$y=\begin{cases} x^2-5x-6, при\ x=6\ или\ x=1 \\ ln|x-1|, если\ 6<x \\ \sqrt[7]{|1-x|}\ , иначе \end{cases}$$

17
$$y=\begin{cases} x^2-3x-4, при\ \ x=4\ или\ \ x=-1 \\ arccos\, x, при\ x\in[0;1] \\ ln\,|2-x|, иначе \end{cases}$$

18
$$y=\begin{cases} x^2-9x-10, при\ x=-1\ или=10 \\ arcctg\ x, при\ x\in[0;1] \\ e^{x+2}, иначе \end{cases}$$

19
$$y=\begin{cases} \dfrac{\sqrt[3]{x-2}}{x}, если\ \ x\in(1;15] \\ arccos\, x,\ если\ \ x=1 \\ lg\, x^4, иначе \end{cases}$$

20
$$y=\begin{cases} x^2-x-2,\ при\ x=-1\ или\ \ x=2 \\ \sqrt[3]{x}-x, если\ \ 2<x \\ \sqrt{e^{x+1}}, иначе \end{cases}$$

21 $$y=\begin{cases} x^2-4x-5, при\ x=-1\ или\ x=5 \\ \ln\sqrt{x+1},\ если\ 0\le x\le 4 \\ \sqrt[5]{|2x-1|},\ иначе \end{cases}$$

22 $$y=\begin{cases} x^2-2x-3,\ при\ x=3\ или\ x=-1 \\ \log_2 x, если\ 5\le x \\ 1+\sqrt[7]{|x+1|}, иначе \end{cases}$$

23 $$y=\begin{cases} x^2-x-12, при\ x=-3\ или\ x=4 \\ \arccos x, при\ x\in[0;1] \\ e^{x-1}, иначе \end{cases}$$

24 $$y=\begin{cases} x^2+8x-20, при\ x=2\ или\ x=-10 \\ \arcsin x, при\ x\in[0;1] \\ 3^x,\ иначе \end{cases}$$

25 $$y=\begin{cases} x^2+3x-10, при\ x=-5\ или\ x=2 \\ \ln\sqrt[3]{x-1}, если\ 2<x\le 10 \\ 2x^5, иначе \end{cases}$$

26 $$y=\begin{cases} x^2+x-6, при\ x=2\ или\ x=-3 \\ \ln\sqrt{|2+x|}, если\ -3\le x \\ \sqrt[5]{|2-x|}, иначе \end{cases}$$

Вычислите значение выражения y при заданном значении x по вариантам:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
x	4	-3	9	-31	-4	-1	0	2	3	9	6	20	-8
	14	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**
x	-24	28	-3	5	-1	10	27	-15	7	-3	4	2	-18

2.13 Программирование циклических процессов обработки информации с параметром

Программирование циклических процессов обработки информации осуществляется с помощью операторов повтора. К ним относятся: цикл с параметром, с предусловием и с постусловием.

Оператор цикла с параметром выполняет повторение действие заданное число раз.

Описание может быть задано одной из двух следующих форм:

– форма 1:
for <параметр>:=<значение 1> *to* <значение 2> *do* <оператор>;
– форма 2:
for <параметр>:=<значение 1> *downto* <значение 2> *do* <оператор>;

<Оператор> называется *телом цикла.* Каждое выполнение тела цикла именуется *итерацией*.

Требования к выполнению:

– выполнение оператора только через начало цикла;

– <значение 1> не должно превышать <значение 2> (форма 1) и наоборот, <значение 1> превышает <значение 2> (форма 2);

– переменная-параметр должна быть переменной перечислимого типа (например, целая, символьная);

– запрещено изменять значение переменной-параметра внутри цикла;

– <значение 1> и <значение 2> могут быть переменными, заранее определенными, причем тип переменных и переменной-параметра должен совпадать;

– можно задавать шаг выполнения цикла только 1 или -1;

– в качестве <оператор> может использоваться составной оператор, если циклически повторяется серия действий;

– прервать выполнение цикла можно двумя способами: увеличить значение параметра цикла до конечного, либо использовать процедуру *break*;

– возврат на новую итерацию цикла осуществляет процедура *continue*, даже если предыдущая итерация не была завершена.

Выполнение: оператор цикла с параметром присваивает переменной-параметру <значение 1>; если оно не превышает <значение 2> (1 форма), то выполняется тело цикла.

Задача 5*. Подсчитайте количество отрицательных и нулевых чисел в заданной последовательности, состоящей из N чисел.*

Программа решения задачи:

```
program pr_5;
var x: real;                              {x – вводимое число}
   N: byte;                               {N – количество чисел}
   i,                                     {i – параметр цикла}
   K_otr, K_nul: integer;                 {K_otr, K_nul – счетчики}
 begin
    write ('Введите количество чисел '); readln (N);
    K_otr:= 0; K_nul:= 0;
    for i:= 1 to N do
                  begin
                        writeln ('Введите число'); readln (x);
                        if x < 0 then K_otr:= K_otr + 1;
                        if x = 0 then K_nul:= K_nul + 1
                  end;
                  writeln ('Количество чисел <0 = ', K_otr,
                        ' количество нулевых чисел = ', K_nul)
 end.
```

Пояснения к программе:

Переменные-счетчики (K_otr, K_nul) относятся к *переменным-накопителям* (в которых происходит накапливание): с выполнением каждой итерации счетчики увеличивают свое значение на 1, а накопители на любое заданное значение.

Переменные-счетчики определяются обязательно целого типа и инициализируются[9] до начала цикла (обнуление).

Задача 6. *Распечатайте чек магазина из отдела канцтоваров, состоящий из названий (на русском языке) N канцтоваров, по заданным номерам товаров.*

Программа решения задачи:

```
program pr_6;
type kanz = (pen, glue, marker, paper, notepad);
    var a: kanz;                          {a – переменная типа kanz}
    N,                                     {N – количество товаров}
    k,                                        {k  – заданный номер}
    i: byte;                                  {i – параметр цикла}
 begin
 write ('Введите количество товаров ');
 readln(N);
 for i:=1 to N do
  begin  write ('Введите номер товара ');
         readln(k);
         a:= veg (k – 1);
         case a of
               kanz (0): writeln (‘Ручка');
               kanz (1): writeln (‘Клей’);
               kanz (2): writeln (‘Маркер');
              kanz (3): writeln (‘Бумага’);
              kanz (4): writeln (‘Блокнот’)
               else writeln (‘Нет такого номера товара’)
         end
   end
 end.
```

Пояснения к программе:

В выражении *veg (k* – 1) номер уменьшается на 1, т. к. нумерация значений перечисляемого типа начинается с нуля.

[9] Процесс инициализации значений переменных-накопителей (счетчиков) заключается в присваивании им начальных (исходных) значений.

Задания для самостоятельного выполнения

1. Даны набор из N целых чисел, вещественные числа A и B (A < B), целое число M. Напишите программу нахождения количества:

1 чисел, кратных *M* и меньших *A*;
2 отрицательных чисел, которые кратны 3;
3 чисел, которые больше или равны *M* и больших *B*;
4 положительных чисел, которые кратны 3;
5 четных чисел, которые меньше *M;*
6 отрицательных чисел из [*A, B*];
7 чисел, не совпадающих с числом *M* и меньших *B*;
8 нечетных чисел, которые меньше *M;*
9 отрицательных чисел, которые кратны 5;
10 чисел из [*A, B*], кратных 5;
11 положительных чисел, которые кратны 5;
12 четных чисел, которые не равны *M*;
13 чисел, кратных 5 и больших *B*;
14 отрицательных чисел, меньших *A*;
15 четных чисел из [*A, B*];
16 положительных чисел, которые больше *M*;
17 чисел, кратных *M* и меньших *B*;
18 нечетных чисел, которые больше *M*;
19 чисел, совпадающих с числом *M* и больших *A*;
20 нечетных чисел, которые не равны *M*;
21 положительных чисел из [*A, B*];
22 четных чисел, которые больше *M*;
23 чисел из [*A, B*], кратных 3;
24 отрицательных чисел, которые меньше *M;*
25 чисел из [*A, B*], кратных *M;*
26 нечетных чисел, больших *A*.

2. Задано описание пользовательских типов данных:

```
type vegetables = (Potato, Eggplant, Corn, Garlic, Kohlrabi,
                   Tomato, Broccoli, Beet, Pumpkin, Lettuce);
```

fruits = (Apple, Orange, Melon, Bananas, Cherry, Coconut, Mango, Watermelon, Pomegranate);

var a: vegetables; b: fruits;

Напечатайте названия овощей и фруктов на русском языке согласно номерам позиций из чека магазина: по значениям переменных a (номер в списке овощей) и b (номер в списке фруктов).

№	*чек*	*№*	*чек*	*№*	*чек*
1	a: 0, 4, 3, 1 b: 7, 3, 5	**10**	a: 7, 5, 9, 8 b: 7, 6, 5	**19**	a: 6, 3, 4, 8 b: 4, 1, 8
2	a: 9, 7, 2, 0 b: 1, 8, 6	**11**	a: 1, 4, 9, 0 b: 2, 3, 5	**20**	a: 1, 2, 9, 0 b: 2, 1, 5
3	a: 4, 7, 5, 0 b: 4, 3, 8	**12**	a: 5, 4, 6, 0 b: 7, 3, 8	**21**	a: 4, 8, 1, 3 b: 3, 6, 5
4	a: 1, 10, 9, 4 b: 2, 0, 5	**13**	a: 1, 2, 4, 9 b: 6, 4, 1	**22**	a: 9, 1, 2, 4 b: 3, 5, 0
5	a: 7, 2, 1, 0 b: 3, 5, 7	**14**	a: 5, 2, 8, 0 b: 3, 2, 7	**23**	a: 2, 6, 9, 1 b: 2, 4, 3
6	a: 4, 5, 9, 1 b: 1, 5, 7	**15**	a: 7, 9, 0, 1 b: 3, 5, 0	**24**	a: 7, 3, 9, 2 b: 2, 5, 8
7	a: 1, 4, 7, 0 b: 0, 3, 7	**16**	a: 1, 4, 7, 0 b: 2, 5, 7	**25**	a: 1, 5, 7, 0 b: 2, 1, 7
8	a: 8, 2, 5, 9 b: 6, 2, 8	**17**	a: 1, 4, 9, 0 b: 5, 7, 8	**26**	a: 1, 7, 8, 9 b: 2, 5, 6
9	a: 2, 4, 6, 1 b: 2, 8, 0	**18**	a: 7, 8, 9, 0 b: 1, 3, 8		

Контрольное задание 3

Дано число b >0 и последовательность чисел x_i, $i = 1…n$.

Напишите программу вычисления суммы $S = \sum_{i=1}^{N} a_i(x_i)$

№	$a_i(x_i)$	№	$a_i(x_i)$	№	$a_i(x_i)$
1	$\frac{\sqrt{b+x_i^3}}{2x_i}$	**10**	$\frac{bx_i}{\sqrt{1+\lvert x_i\rvert}}$	**19**	$\frac{e^{x_i}\sqrt{b}}{3x_i-2}$
2	$\frac{x_i\, e^{x_i-2}}{bx_i+1}$	**11**	$\frac{b+x_i^2}{\sqrt{x_i}+1}$	**20**	$\frac{\sqrt{\lvert 1-2x_i\rvert}}{3bx_i}$
3	$\frac{x_i^2+b}{\ln x_i}$	**12**	$\frac{e^{-x_i}}{\lvert b-x_i\rvert}$	**21**	$\frac{\ln(x_i)}{\sqrt{bx_i}}$
4	$\frac{\lvert 3-x_i\rvert}{2\cdot e^{x_i+1}-b}$	**13**	$\frac{b\sqrt{x_i}}{1-bx_i^3}$	**22**	$\frac{b\sqrt{3x_i}}{2e^{x_i}}$
5	$\frac{4\sqrt{x_i}}{x_i^2-b}$	**14**	$\frac{2\ln x_i}{b\sqrt{x_i}}$	**23**	$b\sqrt{\frac{x_i}{1+x_i}}$
6	$\frac{e^{(x_i-1)}}{bx_i+1}$	**15**	$\sqrt{\frac{2x_i}{1+bx_i}}$	**24**	$\frac{b\sqrt{2x_i}}{1+\ln x_i}$
7	$\frac{bx_i^2-1}{\sqrt{x_i}}$	**16**	$\frac{e^2 x_i}{x_i-\sqrt{4b}}$	**25**	$\frac{bx_i}{\sqrt{\lvert 1-x_i\rvert}}$
8	$\sqrt{\frac{bx_i}{x_i-1}}$	**17**	$\frac{\sqrt{x_i^3}}{bx_i^2}$	**26**	$\frac{2\sqrt{x_i}}{3\sqrt{x_i}-b}$
9	$\frac{be^{-x_i}}{x_i-\ln x_i}$	**18**	$\frac{1}{x_i^2}+\frac{x_i^3}{3b}$		

Вычислите сумму чисел S для n = 7 по вариантам:

№	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	6	5	1	3	2	6	8	-1
2	2	3	2	1	-4	4	-4	6
3	4	2	8	9	5	2	4	3
4	1	1	4	10	-4	-3	2	-8
5	10	9	3	5	1	4	6	8
6	2	2	1	-6	6	-7	-3	-4
7	6	1	5	1	4	5	4	3
8	3	3	6	5	4	2	7	8
9	5	7	1	2	6	7	9	3
10	4	4	2	8	-8	-2	-4	-7
11	3	1	15	7	2	9	5	6
12	8	-4	6	1	-3	-2	4	-8
13	-9	5	4	6	1	2	3	4
14	-4	6	7	3	2	1	5	5
15	7	2	8	5	1	1	2	3
16	3	-8	6	-9	5	6	-6	1
17	-7	4	5	2	3	3	6	7
18	-1	1	-6	-7	5	-5	-7	8
19	10	-2	2	-3	4	-4	-3	3
20	-3	3	-4	1	-2	-7	1	-9
21	4	2	6	7	8	9	3	5
22	-2	4	5	6	1	2	1	3
23	-2	3	6	9	4	3	4	6
24	-5	6	7	2	1	1	2	3
25	-3	2	-5	-1	3	4	-6	8
26	-1	1	3	4	2	6	4	10

2.14 Программирование циклических процессов обработки информации с предусловием

Оператор цикла с предусловием проверяет условие и повторяет действия в зависимости от результата проверки условия.

while <условие> ***do*** <оператор>;

Требования к выполнению:

– <условие> является выражением логического типа и проверяется перед исполнением каждой итерации;

– если повторяется серия действий, то используется составной оператор;

– если <условие> окажется ложным на первой итерации выполнения, то <оператор> может ни разу не выполнится.

Выполнение: до тех пор, пока значение выражения <условие> остается истина, происходит выполнение тела цикла, иначе цикл завершается.

Задача 7. *Вычислите частное сумм кубов положительных и квадратов отрицательных чисел последовательности, если признак завершения ввода чисел – 0.*

Программа решения задачи:

```
program pr_7;
var x,                                   {x – вводимое число}
  Sp,                                    {Sp – сумма положительных чисел}
  So: real;                              {So – сумма отрицательных чисел}
  begin
      Sp:= 0; So:= 0;
      write ('Введите число ');
      readln (x);
      while x <> 0 do begin if x > 0 then Sp:= Sp + sqr (x) * x;
                            if x < 0 then So:= So + sqr (x);
                            writeln ('Введите число'); readln (x)
                      end;
      writeln (‘Частное чисел равно ', Sp / So: 6: 2)
  end.
```

Пояснения к программе:

В программах такого типа инициализация переменных, занятых в условии, осуществляется через ввод первого числа.

Задания для самостоятельного выполнения

1. Дан набор целых чисел и числа: K, A и B (A < B). Признак завершения ввода – 0. Напишите программу вычисления:

1 произведения положительных чисел, кратных 3;
2 суммы чисел, которые больше *K*;
3 удельного веса четных чисел;
4 суммы отрицательных чисел, больших *A;*
5 среднеарифметического значения чисел, меньших *A;*
6 удельного веса чисел, кратных 3;
7 произведения чисел, которые больше или равны *K*;
8 суммы чисел, не совпадающих с числом *K* из [*A, B*];
9 среднеарифметического значения отрицательных чисел;
10 суммы положительных чисел, кратных 5;
11 разности четных и нечетных чисел из [*A, B*];
12 произведения чисел, меньших *K* из [*A, B*];
13 суммы чисел, кратных 3 из [*A, B*];
14 произведения нечетных чисел;
15 удельного веса отрицательных чисел;
16 произведения чисел, не совпадающих с числом *K*;
17 среднегеометрического значения четных чисел;
18 произведения отрицательных чисел, кратных 5;
19 суммы отрицательных чисел из [*A, B*];
20 разности сумм четных и нечетных чисел;
21 удельного веса положительных чисел;
22 суммы нечетных чисел из [*A, B*];
23 среднеарифметического значения чисел из [*A, B*];
24 разности сумм положительных и отрицательных чисел;
25 среднегеометрического значения отрицательных чисел;
26 произведения отрицательных чисел, кратных *K*.

2. Дан набор пар положительных чисел $x_1, y_1, x_2, y_2, \ldots$. Признак завершения ввода – первое введенное отрицательное число. Напишите программу вычисления суммы

$$S = \sum_{i=1} b_i(x_i, y_i).$$

№	$b_i(x_i, y_i)$	№	$b_i(x_i, y_i)$	№	$b_i(x_i, y_i)$
1	$\frac{2x_i + 1}{lg(y_i^2 + 2)}$	**10**	$\frac{y_i + 1}{1 + \log_2(x_i + 1)}$	**19**	$\frac{2^{y_i} - 1}{x_i^2 + 2}$
2	$\frac{\sqrt[5]{x_i^3 + 5}}{x_i + y_i}$	**11**	$\frac{x_i + 3}{lg(y_i + 1)}$	**20**	$\frac{lg(1 + y_i)}{x_i - 1}$
3	$\frac{y_i + x_i}{\sqrt[3]{2x_i + 1}}$	**12**	$\frac{\sqrt[3]{x_i + 1}}{3 - 2y_i}$	**21**	$\frac{x_i + 1}{lg(y_i^2 + 1)}$
4	$\frac{ln(x_i + 1)}{lg\left(\sqrt{y_i^3} + 1\right)}$	**13**	$\frac{\lvert x_i - y_i \rvert}{\log_5(3x_i + 1)}$	**22**	$\frac{x_i - 1}{\sqrt[3]{y_i + 1}}$
5	$\frac{\log_3(1 + x_i)}{ln(y_i + 1)}$	**14**	$\frac{2^{x_i} - 1}{2y_i^2 + 1}$	**23**	$\frac{y_i - 1}{\sqrt[3]{1 + x_i^2}}$
6	$\frac{\log_7 x_i^3}{y_i + 1}$	**15**	$\frac{\log_2(x_i + 2)}{y_i - 1}$	**24**	$\frac{2^{y_i} - 1}{2x_i^3 + 1}$
7	$\frac{x_i + 1}{lg\sqrt{y_i + 1}}$	**16**	$\frac{\sqrt[3]{y_i^2 + 3}}{1 + 2x_i}$	**25**	$\frac{y_i^2 + 1}{3^{x_i}}$
8	$\frac{\sqrt[3]{y_i^3 + 2}}{x_i + y_i}$	**17**	$\frac{y_i - 1}{\log_2(x_i^2 + 1)}$	**26**	$\frac{x_i^2 - 5x_i + 6}{lg(y_i + 1)}$
9	$\frac{\lvert y_i - x_i \rvert}{\sqrt[5]{y_i^2 + 1}}$	**18**	$\frac{\sqrt[3]{y_i - 1}}{1 - ln^2 x_i}$		

Контрольное задание 4

Дан набор пар положительных чисел $x_1, y_1, x_2, y_2, \ldots$. Признак завершения ввода – 0. Напишите программу вычисления произведения $P = \prod_{i=1} c_i(x_i, y_i)$.

№	$c_i(x_i, y_i)$	№	$c_i(x_i, y_i)$	№	$c_i(x_i, y_i)$
1	$\frac{\sqrt[3]{x_i + 1}}{3 - 2y_i}$	**10**	$\frac{x_i + 1}{lg(y_i^2 + 1)}$	**19**	$\frac{y_i + x_i}{\sqrt[3]{2x_i + 1}}$
2	$\frac{\lvert x_i - y_i \rvert}{log_5(3x_i + 1)}$	**11**	$\frac{x_i - 1}{\sqrt[3]{y_i + 1}}$	**20**	$\frac{ln(x_i + 1)}{lg\left(\sqrt{y_i^3} + 1\right)}$
3	$\frac{y_i + 1}{1 + log_2(x_i + 1)}$	**12**	$\frac{2^{y_i} - 1}{x_i^2 + 2}$	**21**	$\frac{2x_i + 1}{lg(y_i^2 + 2)}$
4	$\frac{x_i + 3}{lg(y_i + 1)}$	**13**	$\frac{lg(1 + y_i)}{x_i - 1}$	**22**	$\frac{\sqrt[5]{x_i^3 + 5}}{x_i + y_i}$
5	$\frac{2^{x_i} - 1}{2y_i^2 + 1}$	**14**	$\frac{y_i - 1}{\sqrt[3]{1 + x_i^2}}$	**23**	$\frac{log_3(1 + x_i)}{ln(y_i + 1)}$
6	$\frac{y_i - 1}{log_2(x_i^2 + 1)}$	**15**	$\frac{x_i^2 - 5x_i + 6}{lg(y_i + 1)}$	**24**	$\frac{\sqrt[3]{y_i^3 + 2}}{x_i + y_i}$
7	$\frac{\sqrt[3]{y_i - 1}}{1 - ln^2 x_i}$	**16**	$\frac{y_i^2 + 1}{3^{x_i}}$	**25**	$\frac{\lvert y_i - x_i \rvert}{\sqrt[5]{y_i^2 + 1}}$
8	$\frac{log_2(x_i + 2)}{y_i - 1}$	**17**	$\frac{2^{y_i} - 1}{2x_i^3 + 1}$	**26**	$\frac{log_7 x_i^3}{y_i + 1}$
9	$\frac{\sqrt[3]{y_i^2 + 3}}{1 + 2x_i}$	**18**	$\frac{x_i + 1}{lg\sqrt{y_i + 1}}$		

Вычислите произведение чисел P по вариантам:

№	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4	x_5	y_5
1	2,1	1,4	5,2	5,6	4,1	6,3	7,1	3,2	0	0
2	1,8	1,7	2,4	3,0	7,0	2,1	8,0	5,7	0	0
3	5,0	6,8	6,7	1,4	3,3	8,6	6,5	9,8	0	0
4	3,4	3,4	6,6	7,5	7,5	4,5	5,4	2,0	0	0
5	5,2	5,9	7,7	8,6	1,2	1,6	6,6	7,8	0	0
6	4,6	4,0	3,8	3,1	10,1	2,8	2,2	3,9	0	0
7	2,1	7,3	1,9	9,2	3,4	3,9	4,5	1,4	0	0
8	8,7	9,2	3,0	5,0	4,5	2,0	6,6	7,0	0	0
9	1,3	2,9	1,4	3,5	2,6	4,3	5,7	6,4	0	0
10	2,5	3,1	6,6	9,7	4,6	3,5	4,8	8,6	0	0
11	5,6	6,3	7,4	2,2	1,6	1,6	2,5	3,2	0	0
12	3,7	2,4	5,3	1,5	3,0	4,0	6,9	8,2	0	0
13	1,2	1,5	3,2	4,2	2,8	6,5	4,3	10,5	0	0
14	2,9	6,0	8,2	1,8	6,7	5,3	1,6	3,8	0	0
15	4,3	4,1	4,1	6,0	2,8	3,6	2,9	1,1	0	0
16	5,6	2,9	4,5	3,4	4,4	2,1	8,2	9,6	0	0
17	4,0	3,3	2,0	8,2	1,9	1,3	4,9	10,1	0	0
18	1,7	4,3	6,3	8,6	10,3	9,8	3,1	5,7	0	0
19	6,2	7,5	3,2	4,4	2,4	2,2	1,7	6,9	0	0
20	4,2	5,2	4,1	3,8	6,5	1,5	5,3	1,0	0	0
21	4,1	2,1	7,8	8,2	3,6	3,3	6,7	5,7	0	0
22	6,7	7,3	9,4	3,6	5,3	7,5	1,3	2,3	0	0
23	8,0	2,2	4,8	7,4	4,1	4,6	2,2	8,2	0	0
24	2,6	9,4	5,9	6,6	3,4	1,9	5,4	7,5	0	0
25	3,7	2,5	4,6	8,1	8,4	4,0	6,8	1,6	0	0
26	1,3	2,7	3,0	4,3	9,6	5,3	4,6	6,7	0	0

2.15 Программирование циклических процессов обработки информации с постусловием

Оператор цикла с постусловием осуществляет повтор действия в зависимости от результата проверки заданного условия.

repeat <оператор> ***until*** <условие>;

Требования к выполнению:

– <условие> является выражением логического типа и проверяется после исполнения каждой итерации;

– для повторения серии действий используется составной оператор;

– тело цикла выполнится обязательно хотя бы один.

Выполнение тела цикла происходит до тех пор, пока значение выражения <условие> остается ложь, иначе цикл завершается.

Задача 8. *Известны данные (%) об успеваемости студентов N групп курса. Определите средний процент успеваемости на курсе.*

Программа решения задачи:

```
program pr_8;
var x,                          {x – успеваемость в группе (%)}
    sum: real;                  {sum – суммарная успеваемость}
    N: byte;                    {N – количество групп на курсе}
    i: integer;                 {i – параметр цикла}
begin
   sum:= 0;
   write (‘Введите количество групп курса‘); readln (N);
   for i:= 1 to N do
     begin
        repeat
           write (‘Введите успеваемость в ‘, i, ‘ группе (%)’);
           readln (x);
           if x<= 0 then writeln (‘Ошибка ввода’)
                    else  sum:= sum + x
        until (x <= 100) and (x > 0)
```

```
    end;
  writeln (‘sum = ‘, sum / N : 5: 2, ‘%’)
end.
```

Пояснения к программе:

Контроль корректности ввода значений переменных очень часто осуществляется с помощью цикла с постусловием.

Задача 9. *Напишите программу вычисления суммы ряда с точностью ε = 0,001, общий член которого равен:* $a_n = \frac{(-1)^{(n-1)} n}{\sqrt{n+1}}$.

Программа решения задачи:

```
program pr_9;
const E = 0.001;                          {E – заданная точность}
var S, z: real;                           {S – сумма ряда; z – элемент ряда}
n, y: integer;                            {n – номер элемента; y – знак элемента}
begin
    S:= 0; n:= 1;
    repeat
        if  n mod 2 = 0 then y:= -1 else y := 1;
        z:= y * n / sqrt (n + 1);
        S:= S + z;
        n:= n + 1
    until (z < E);
    write ('Сумма ряда равна ', S – z : 6 : 3)
end.
```

Пояснения к программе:

Числитель дроби является знакопеременным: при четном *n* числитель равен *-n*, иначе *n*.

Формулировка: «вычислить сумму ряда с определенной точностью (*ε* = 0,001)» означает, что ответ будет отличаться от истинного значения не более чем на 0,001. Очередное слагаемое (*z*) сравнивается с точностью (*ε*): если точность достигнута, то последнее слагаемое было лишним, поэтому его не учитывают *(S – z)*.

Задания для самостоятельного выполнения

1. Известны данные об итогах сдачи тестирования (%) студентов в N группах курса и заданы значения уровней подготовки (%): A1 (начальный), A2 (средний), A3 (высший). Определите:

1 номер группы, в которой наибольшее количество студентов, превысивших начальный уровень;
2 количество студентов в первой группе, превысивших высший уровень;
3 номера студентов, показавших наилучший результат;
4 номер группы, в которой наибольшее количество студентов показали результат в интервале [*A*2; *A*3];
5 количество студентов на курсе, превысивших начальный уровень;
6 номер группы, в которой наименьшее количество студентов, превысивших высший уровень;
7 количество студентов на курсе, показавших результат в интервале [*A*1; *A*3];
8 номер группы, в которой наибольшее количество студентов, превысивших средний уровень;
9 количество студентов на курсе, превысивших средний уровень;
10 номера студентов, показавших наихудший результат;
11 количество студентов в *N*-ой группе, превысивших высший уровень;
12 номер группы, в которой наименьшее количество студентов, превысивших начальный уровень;
13 количество студентов на курсе, показавших результат в интервале [*A*1; *A*2];
14 количество студентов на курсе, превысивших высший уровень;
15 номер группы, в которой наибольшее количество студентов показали результат в интервале [*A*1; *A*2];
16 количество студентов в первой группе, превысивших

начальный уровень;

17 номер группы, в которой наименьшее количество студентов, превысивших средний уровень;

18 количество студентов в *N*-ой группе, превысивших начальный уровень;

19 номер группы, в которой наибольшее количество студентов показали результат в интервале [*A*1; *A*3];

20 количество студентов в первой группе, превысивших средний уровень;

21 количество студентов на курсе, не сдавших тестирование;

22 количество студентов на курсе, показавших результат в интервале [*A*2; *A*3];

23 номер группы, в которой наибольшее количество студентов, не сдавших тестирование;

24 средний процент тестирования в каждой из групп;

25 количество студентов в *N*-ой группе, превысивших средний уровень;

26 номер группы, в которой наибольшее количество студентов, превысивших высший уровень.

2. Дан набор чисел $x_1, x_2,\ldots$. Признак завершения ввода – 0. Напишите программу вычисления суммы выражений.

0	$\frac{ln\lvert x_i\rvert - 1}{\sqrt{1+x_i^2}}$	**1**	$\frac{x_i^2 - 2}{lg\lvert x_i\rvert - 1}$	**2**	$\frac{x_i}{\sqrt{1+ln\lvert x_i\rvert}}$
3	$\frac{\sqrt{\lvert x_i^3\rvert}+1}{x_i - e^{x_i}}$	**4**	$\frac{\sqrt{\lvert x_i\rvert}-1}{1-ln\lvert x_i\rvert}$	**5**	$\frac{e^{x_i}-1}{2\sqrt{x_i^2+1}}$
6	$\frac{\sqrt{\lvert x_i^3\rvert}+2}{x_i - 2}$	**7**	$\frac{lg\lvert 1+x_i\rvert}{x_i - 1}$	**8**	$\frac{\lvert\sqrt{1-x_i}\rvert}{ln\lvert x_i\rvert+1}$
9	$\frac{lg\lvert x_i-1\rvert}{x_i-3}$				

Контрольное задание 5

Напишите программу вычисления суммы ряда с точностью ε = 0,001, общий член которого равен:

1	$\frac{\sqrt[3]{n+1}}{3-2n}$	**10**	$\frac{(-1)^{n+1}}{n-\lg(n+1)}$	**19**	$\frac{\lg(n+2)}{2n+1}$
2	$\frac{(-1)^{n+1}n^2}{1+\log_5(n+1)}$	**11**	$\frac{(-1)^{n-1}}{\sqrt[3]{n^3+1}}$	**20**	$\frac{(-1)^{n-1}}{\lg\left(\sqrt{n^3}+1\right)}$
3	$\frac{n^2+1}{3^n}$	**12**	$\frac{n^2+1}{4^n(n+2)}$	**21**	$\frac{(-1)^{n-1}}{\sqrt[3]{2n+1}}$
4	$\frac{n+3}{n^2\ln(n+1)}$	**13**	$\frac{\lg(1+n)}{(-1)^{n+1}}$	**22**	$\frac{\sqrt[5]{n+1}}{n+2}$
5	$\frac{2n+1}{3^n-1}$	**14**	$\frac{\sqrt[3]{1+n^2}}{(-1)^{n-1}}$	**23**	$\frac{(-1)^{n-1}}{\log_3(1+n)}$
6	$\frac{(-1)^{n-1}}{9\log_2(n+1)}$	**15**	$\frac{n^2-5n+6}{\lg(n+1)}$	**24**	$\frac{\sqrt[3]{n+2}}{n+1}$
7	$\frac{1-\ln n}{\sqrt[3]{2n-1}}$	**16**	$\frac{(-1)^{n-1}}{\log_5(3n+1)}$	**25**	$\frac{(-1)^{n+1}}{\sqrt[5]{n^2+1}}$
8	$\frac{\log_2(n+2)}{(-1)^{n-1}}$	**17**	$\frac{2n-1}{2^n+1}$	**26**	$\frac{(-1)^{n+1}}{\log_{100}(100n^3)}$
9	$\frac{\sqrt[3]{n+1}}{1+2n}$	**18**	$\frac{(-1)^{n+1}}{\lg\sqrt{n+1}}$		

Контрольные вопросы к главе 2

1. Какими отличительными особенностями обладает язык программирования Паскаль?
2. Что такое алгоритмическое программирование?
3. Какова структура программы?
4. Что представляет собой алфавит языка программирования?
5. Как соотносятся понятия синтаксиса и семантики языка программирования?
6. По каким правилам создаются идентификаторы?
7. В чем различие понятий синтаксиса и семантики языка программирования?
8. Как соотносятся понятия: имя, тип и значение данных?
9. Что составляет структуру типов данных?
10. Какие простые стандартные типы данных определены в языке Паскаль?
11. Какие новые типы данных может определить пользователь?
12. Что такое выражение? Как определить тип выражения?
13. Что такое идентификатор?
14. Каков состав разделов описаний программы?
15. Какие операторы можно отнести к операторам ввода данных с клавиатуры?
16. Какие операторы можно отнести к операторам вывода данных на экран дисплея?
17. Какие операторы считаются простыми операторами?
18. Что такое структурное программирование?
19. Какие операторы реализуют разветвление в программах?
20. Какие операторы реализуют множественное ветвление в программах?
21. Какой оператор реализует цикл с параметром?
22. Какой оператор реализует цикл с предусловием?
23. Какой оператор реализует цикл с постусловием?

Ответы к контрольным заданиям

Ответы к контрольному заданию 1

1) 0,89	2) 0,03	3) 2,53	4) 7,38	5) 0,06
6) 0,55	7) 2,73	8) 0,16	9) 18,02	10) 5,44
11) 0,55	12) 0,08	13) 6,58	14) 0,17	15) 15,21
16) 0,02	17) 0,12	18) 0,44	19) 0,78	20) 0,02
21) 7,13	22) 2,52	23) 0,01	24) 5,29	25) 0,06
26) 1,07				

Ответы к контрольному заданию 2

1) -8	2) 0,8	3) 2	4) 4	5) -10
6) 0,7	7) 1	8) 53,1	9) 0,6	10) 2
11) 3,6	12) 3	13) 0,5	14) 5	15) -81
16) 1,4	17) 1,1	18) 2,7	19) 0,2	20) -24
21) 2	22) 2,8	23) -1,3	24) 81	25) 16
26) 1,4				

Ответы к контрольному заданию 3

1) 5,91	2) 30,17	3) 138,75	4) 26,85	5) 4,69
6) 12,29	7) 268,87	8) 14,13	9) 2,49	10) 9,9
11) 106,91	12) 193,48	13) 1,23	14) 2,02	15) 3,61
16) 71,2	17) 0,51	18) 130,83	19) 32,09	20) 0,32
21) 2,39	22) 1,86	23) 12,64	24) 44,35	25) 13,31
26) 3,93				

Ответы к контрольному заданию 4

1) -3,37	2) 0,97	3) 11,32	4) 32,46	5) 0,12
6) 0,11	7) 23,57	8) 0,34	9) 0,05	10) 202,38
11) 11,18	12) 0,13	13) 0,10	14) 1,02	15) 0,78
16) 126,53	17) 0,55	18) 30,44	19) 323,66	20) 3,69
21) 761,46	22) 0,01	23) 1,88	24) 0,09	25) 12,29
26) 169,26				

Ответы к контрольному заданию 5

1) 1,260	2) 0,699	3) 1,999	4) 9,134	5) 2,579
6) 0,111	7) 1,213	8) 1,585	9) 32,143	10) 1,431
11) 0,794	12) 0,292	13) 0,301	14) 1,260	15) 1,253
16) 1,161	17) 2,574	18) 6,644	19) 6,158	20) 3,322
21) 0,693	22) 21,977	23) 1,585	24) 91,769	25) 0,871
26) 0,094				

ЗАКЛЮЧЕНИЕ

Одним из главных достоинств этого пособия является краткое и в то же время, достаточно исчерпывающее изложение ответов на любой поставленный вопрос в доступном для восприятия стиле.

В примерной программе дисциплины «Информатика» определены задачи профессиональной деятельности. В том числе к ним относятся:

– сформировать навыки работы в интегрированной вычислительной системе и среде программирования;

– сформировать навыки разработки и отладки программ, получения и анализа результатов с использованием языка высокого уровня.

Учебное пособие призвано помочь в решении поставленных задач:

– рассмотрена элементарная база понятий в области программирования;

– изложены основные принципы работы в среде программирования *Turbo Pascal*;

– приведены упражнения для самостоятельного выполнения в среде *Turbo Pascal*;

– представлено структурированное изложение теории по основам программирования на языке Паскаль, которое сопровождено примерами программ с пояснениями;

– приведены задания для самостоятельного выполнения по вариантам;

– приведены контрольные задания для самостоятельного выполнения по вариантам с ответами.

Все программы из данного учебного пособия проверены в среде программирования *Turbo Pascal for Windows* 7 и являются работоспособными.

СПИСОК ЛИТЕРАТУРЫ

1.	Анищик Т. А. Алгоритмизация и программирование на языке Паскаль [Текст]: учеб. пособие / Т. А. Анищик. – Краснодар: КубГАУ, 2016. – 160 с. – Режим доступа: http://edu.kubsau.ru/file.php/118/Uchebnoe_posobie_2016.pdf
2.	Анищик Т. А. Язык программирования Паскаль в вопросах и ответах [Текст]: учеб. -метод. пособие / Т. А. Анищик. – Краснодар: КубГАУ, 2001. – 96 с.
3.	Анищик Т. А. Практикум по математическим и логическим основам информатики [Текст]: учеб. -метод. пособие / Т. А. Анищик. – Краснодар: КубГАУ, 2013. – 108 с. – Режим доступа: http://edu.kubsau.ru/file.php/118/Praktikum_po_MLOI_2013g.pdf
4.	Грогоно П. Программирование на языке Паскаль [Текст]: учебник / П. Грогоно. – М.: Мир, 1982. – 384 с.
5.	Дейкстра Э. Дисциплина программирования [Текст] / Э. Дейкстра. – М.: Мир, 1978. – 275 с.
6.	Йенсен К. Паскаль. [Текст]: руководство для пользователя / К. Йенсен, Н. Вирт. – М.: Финансы и статистика, 1989. – 255 с.
7.	Основы программирования и алгоритмические языки [Текст]: учебник / В. Г. Баула, Н. Д. Васюкова, В. В. Тюляева, П. В. Уманец. – М.: Энергоатомиздат, 1991. – 398 с.
8.	Новиков Ф. А. Технологические подходы к разработке программного обеспечения [Электронные данные]: учеб. -метод. пособие / Ф. А. Новиков. – СПб 2007. – 137 с. – Режим доступа: http://books.ifmo.ru/file/pdf/427.pdf
9.	Пильщиков В. Н. Сборник упражнений по языку Паскаль [Текст]: учеб. пособие / В. Н. Пильщиков. – М.: Наука, 1989. – 160 с.

ПРИЛОЖЕНИЕ 1

Назначение специальных символов

Знак	*Назначение*	*Знаки*	*Назначение*
Знаки пунктуации			
‘	апостроф, одиночная кавычка	{ }	скобки комментария
;	разделение операторов и объявлений	( )	обозначение параметров, обычные скобки
:	отделение переменной от типа	[ ]	обозначение индексов массивов и элементов множеств
,	разделение элементов списка	(* *)	альтернатива скобок комментария
$	знак шестнадцатеричного числа или директива компилятора	(. .)	альтернатива обозначения индексов массивов и элементов множеств
=	отделение идентификатора типа от описания типа или константы от ее значения	.	конец программы, отделение целой части от дробной, разделение полей записи
#	признак кода числа	:=	знак присваивания
..	разделение границ диапазона		
Знаки арифметических действий и операций отношения			
+	сложение	<, >	меньше, больше
–	вычитание	=, <>	равно, не равно
*	умножение	<=	меньше или равно
/	деление	>=	больше или равно
div	целочисленное деление	*shr*	сдвиг битов вправо
mod	деление с остатком	*shl*	сдвиг битов влево

ПРИЛОЖЕНИЕ 2

Список зарезервированных слов
языка программирования Паскаль

Название	*Назначение*
and	логическое И
array	объявление массива
assembler	директива объявления подпрограммы, написанной на языке ассемблера
begin	начало тела программы, подпрограммы или составного оператора
case	оператор выбора
const	задание константы
div	получение целой части от деления
do	признак начала оператора цикла
downto	указание на просмотр элементов цикла с параметром от большего к меньшему значению
else	альтернативная часть в условном операторе
end	конец тела программы, подпрограммы или составного оператора
external	директива позволяет использовать подпрограммы, написанные на языке ассемблера и скомпилированные отдельно
far	директива сообщает компилятору, что нужно формировать «дальний» адрес
for	начало безусловного цикла
forward	директива включения одной подпрограммы в другую
function	начало объявления функции
goto	оператор перехода
if	начало условного оператора
in	проверка принадлежности множеству

inline	директива включения в тело программы части, написанной на языке ассемблера
interrupt	директива предназначена для процедур, обрабатывающих прерывания
label	объявление метки
mod	получение остатка от деления
nil	пустой указатель
not	логическое НЕ
object	объявление объекта
of	тип компонент в определении массива
or	логическое ИЛИ
packed	признак «упакованного массива»
procedure	начало объявления процедуры
program	начало объявления программы
record	объявление записи
repeat	начало цикла с постусловием
set	объявление множества
shl	сдвиг битов влево
shr	сдвиг битов вправо
string	объявление строки
then	продолжение условного оператора
to	указание на просмотр элементов от меньшего к большему значению в цикле с параметром
type	описание типа
unit	начало объявления модуля
until	завершение цикла с постусловием
uses	подключение библиотеки
var	объявление переменных
while	начало цикла с предусловием
with	начало оператора присоединения
xor	логическое исключающее ИЛИ

ПРИЛОЖЕНИЕ 3

Функции работы
с простыми стандартными типами данных

Имя функции	*Назначение*	*Пример использования*
Целочисленные функции		
ord (x)	– порядковый номер значения *x* в множестве, определенном типом x	*ord('a') = 65*
dec (x)	– уменьшение *x* на 1	*dec(10)=9*
inc (x)	– увеличение *x* на 1	*inc(10)=11*
trunc (x)	– получение целой части вещественного числа *x*	*trunc(5.85) = 5*
round (x)	– округление до целого вещественного числа *x*	*round(5.85) = 6*
Randomize	– задает начальное значение последовательности, образованной *Random*	*Randomize;* *x:= Random(20);* *x ∈ [0..20]*
Random (i)	– генерирует значение случайного числа из диапазона *0..i.*	
Вещественные функции		
abs (x)	– абсолютная часть числа *x*	*abs(2) = 2*
arctan (x)	– арктангенс числа *x*	*arctan(0) = 0*
cos (x)	– косинус числа *x*	*cos(pi) = 1*
sin (x)	– синус числа *x*	*sin(pi) = 0*
ln (x)	– натуральный логарифм числа *x*	*ln(1) = 0*
exp (x)	– экспонента e^x	*exp(0) = 1*

sqr (x)	– возведение в квадрат числа *x*	*sqr(5) = 25*
sqrt(x)	– квадратный корень числа *x*	*sqrt(16) = 4*
frac (x)	– дробная часть *x*	*frac(3.14) = 0.14*
int (x)	– целая часть числа *x*	*int(3.64) = 3*
Символьные функции		
pred (x)	– выдает предыдущее значение	*pred('c') = 'b'*
succ (x)	– выдает последующее значение	*succ('c') = 'd'*
ord (x)	– выдает численное значение кода	*ord('9') ord('1') = 8*
chr (x)	– выдает символ по числовому коду	*chr(70) = 'f'*
upcase (x)	– преобразует строчную букву в прописную	*upcase('a') = 'a'*
Логические функции и операции		
odd (x)	– проверка *x* на четность	*odd(3) = true;* *odd(4) = false;*
not	– операция логического отрицания	*x := true;* *not x = false;*
and	– операция логического умножения (конъюнкции)	*x := true; y :=false;* *x and y = false;*
or	– операция логического сложения (дизъюнкции)	*x = true; y :=false;* *x or y = true;*
xor	– исключающее ИЛИ (строгая дизъюнкция)	*x = true; y :=false;* *x xor y = true;*

Учебное издание

Анищик Татьяна Алексеевна

ОСНОВЫ АЛГОРИТМИЧЕСКОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ПАСКАЛЬ. Часть 1

Учебное пособие

В авторской редакции

Компьютерная верстка – Т. А. Анищик

Подписано в печать_____. Формат 60×84 1/16,

Усл. печ. л. – 5,2. Тираж ____ экз. Заказ №.___

Типография Кубанского государственного
аграрного университета
350044 г. Краснодар, ул. Калинина, 13