

Федеральное агентство по образованию

**ПСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ**

О.А. Полетаева

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ ПАСКАЛЬ

Методические указания

по выполнению курсовой работы
для студентов очной формы обучения специальности
119601 Автомобили и автомобильное хозяйство

Псков
2009

Содержание

ОБЩИЕ ПОЛОЖЕНИЯ.....	3
Введение.....	3
Язык, оболочка и интегрированная среда разработки Турбо Паскаль.....	4
Интегрированная инструментальная оболочка Турбо Паскаль.....	5
Отладка и выполнение программы.....	8
СПИСОК ВОПРОСОВ ПО ТЕОРЕТИЧЕСКОЙ ЧАСТИ.....	15
ПОРЯДОК ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ.....	17
ЧАСТЬ 1. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЛИНЕЙНЫХ И ВЕТВЯЩИХСЯ АЛГОРИТМОВ.....	19
Задание 1.1. Программирование формул.....	19
Задание 1.2. Ветвящиеся алгоритмы.....	24
ЧАСТЬ 2. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ.....	31
Задание 2.1. Циклы с известным числом повторений.....	31
Задание 2.2. Двойные и кратные циклы.....	37
ПРИЛОЖЕНИЕ А.....	42
ПРИЛОЖЕНИЕ Б.....	45
Список литературы.....	49

ОБЩИЕ ПОЛОЖЕНИЯ

Введение

Решение задачи на ЭВМ с составлением программы состоит из четырех этапов:

1. Постановка задачи.
2. Составление алгоритма.
3. Составление программы.
4. Ввод и отладка программы.

То есть, прежде чем приступить к непосредственному составлению программы, – написанию последовательности операторов языка, – необходимо отчетливо представить себе ход процесса вычислений, ту последовательность действий, которую должна реализовать программа. Первый этап не поддается строгой формализации и может быть достаточно сложным для больших задач, но в контрольных работах постановка задачи приводится в самом задании.

Строгое представление последовательности действий, то есть алгоритм, наиболее удобно изображать графически с помощью блок-схем или граф-схем, хотя его можно описать и другими способами, например словесно. На этапе обучения программированию использование блок-схем является обязательным.

Составление программы выполняется на каком-либо языке программирования. В данном курсе используется язык высокого уровня Паскаль. При составлении программы необходимо строго придерживаться правил записи программы, которые изложены в литературе, например, приведенной в списке [1-6].

Для ввода и отладки программы используется система программирования Турбо Паскаль, краткие сведения о которой приводятся далее. Этот этап включает и тестирование программы, то есть проверку ее работоспособности при самых разнообразных условиях эксплуатации и вводимых данных.

При профессиональном программировании существует и пятый этап – сопровождение программы. Он заключается в исправлении замеченных в ходе работы с программой ошибок и изменениях по улучшению эксплуатационных свойств программы. Для больших программных комплексов (например, операционных систем) это один из самых трудоемких этапов, наряду с тестированием программы.

Язык, оболочка и интегрированная среда разработки Турбо Паскаль

Хотя язык Паскаль является относительно старым языком программирования, созданным Никлаусом Виртом в 1968 году специально для обучения студентов программированию, но с появлением персональных компьютеров он получил широкое распространение не только в образовательной сфере, но и при решении различных прикладных задач. В 1983 году появилась первая версия Паскаля, предназначенная для IBM-совместимых компьютеров. С тех пор среда Турбо Паскаль и непосредственно язык Паскаль непрерывно совершенствовались фирмой **Borland International**. В 1992 году была представлена очередная версия системы программирования – Турбо Паскаль 7.0, работа с которой и будет здесь рассмотрена. В настоящее время широко применяется среда разработки **Delphi** для операционных систем **Windows**. В ней используется тот же язык Паскаль с дополнительными, но не принципиальными возможностями.

После того, как программа составлена, ее необходимо ввести в компьютер. Здесь мы и сталкиваемся с инструментальным пакетом программ Турбо Паскаль. Этот пакет содержит не только компилятор (или транслятор) с языка Паскаль, но и редактор текста, инструментальную оболочку, отладчик, обширные библиотеки программ под **DOS** и **Windows**, драйвера видеоадаптеров и памяти, и многое другое, в частности примеры программ с использованием методов объектно-ориентированного программирования.

Для связи основных из этих программ в единое целое, создания удобного и наглядного интерфейса предназначена **интегрированная инструментальная оболочка**, в дальнейшем именуемая ИИО. Весь же набор программ называется системой программирования, инструментальной системой, **интегрированной средой разработки** (**Integrated Development Environment, IDE**) или **просто Турбо-средой**.

Язык Паскаль, используемый в Турбо-среде, является расширением стандартного языка программирования Паскаль, поэтому все программы, написанные на языке Паскаль, будут выполняться и в среде Турбо Паскаль, хотя обратное утверждение несправедливо. То есть соблюдается полная преемственность по принципу «сверху-вниз».

Для того, чтобы отличить стандартный язык программирования Паскаль от его расширения, последний называют **расширением Паскаля в Турбо-среде** или **просто Турбо Паскалем**.

Интегрированная инструментальная оболочка Турбо Паскаль

Вызов ИИО в операционных системах **Windows** осуществляется запуском на выполнение файла **TURBO.EXE**, работающего в окне **DOS**. Для работы в окне **DOS** так же можно использовать файловый менеджер (оболочку операционной системы), например **Norton Commander**, **FAR** и другие.

Файл **TURBO.EXE** может находиться в любом месте, но при начальной установке вся инструментальная система помещается в папку **BP (Borland Pascal)**, а файл во вложенную папку **BIN**. То есть путь доступа к файлу ИИО может быть, например, таким: **D:\BP\BIN\TURBO.EXE**. В любом случае при выполнении контрольных работ в компьютерном классе преподаватель должен сообщить о его расположении, а так же где находятся папки для сохранения текстов программ.

Окно **DOS** открывается либо на весь экран (полноэкранный режим), либо в окне **Windows** (оконный режим). Переход между этими режимами выполняется комбинацией клавиш **<Alt+Enter>**.

Примечание. В англоязычной литературе для комбинации клавиш, позволяющих быстро вызвать часто используемую последовательность действий, существует общепринятый термин **Hot Keys** – горячие клавиши. В Приложении А, табл. 5, приводятся все команды ИИО и соответствующие им «горячие клавиши».

После запуска файла **TURBO.EXE** основной экран ИИО, состоящий из трех частей: строки меню (под заголовком окна), рабочей зоны и строки состояния (в самом низу), приведен на рис. 1.

При начальном запуске ИИО в рабочей зоне открыто одно окно редактирования программы с номером **1** (в правом верхнем углу рабочего поля) и с заголовком **NONAME00.PAS**. В дальнейшем, после сохранения программы на диск, стандартный заголовок заменится на имя программы, данное ей при записи. Если рабочая зона пустая, то необходимо создать новое окно командой **File/New**, как представлено на рис. 2.

В окне редактирования набирается, просматривается и редактируется текст программы с помощью встроенного редактора текста. Для его освоения можно использовать приложение Б, табл. 6, в котором дана краткая справка по командам редактора, либо использовать справочную подсистему **Help** ИИО.

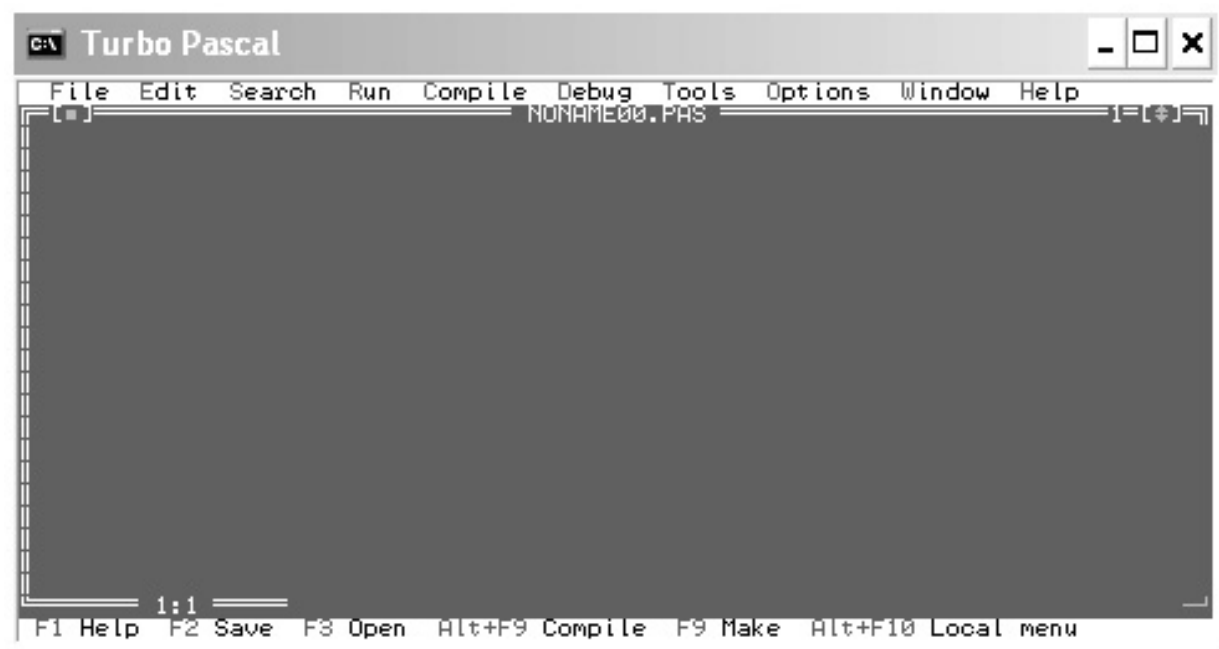


Рис. 1. Вид окна ИИО после запуска файла Turbo.exe в оконном режиме

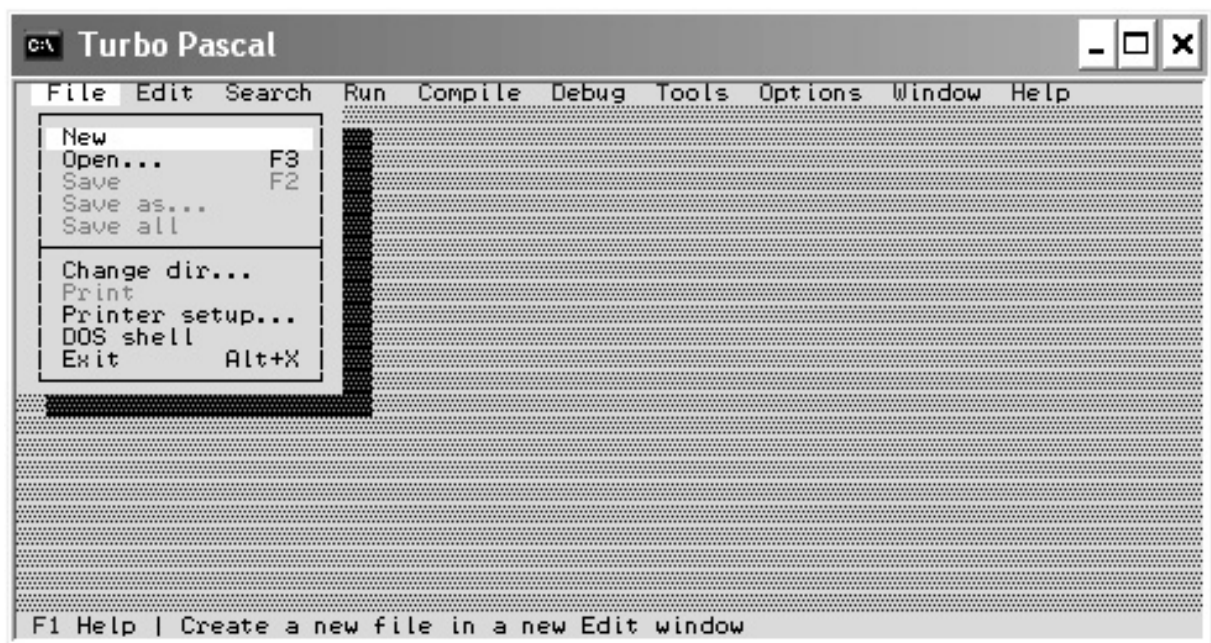


Рис. 2. Создание нового окна программы

Мигающий курсор указывает то место на экране, в котором будет появляться или редактироваться текст. Его местоположение указывается в левом нижнем углу в виде координат <строка>:<столбец>. Ввод каждой новой строки заканчивается нажатием клавиши *Enter*. Компилятор не различает прописные и строчные буквы, поэтому все равно, в каком регистре набираются латинские буквы, так, следующие строки будут эквивалентными:

Program A;
PROGRAM a;
PrOgRaM a;

Набирая текст, особое внимание необходимо обращать на точное воспроизведение всех знаков: точек, точек с запятой, апострофов, пробелов, так как компилятор очень чувствителен к мелочам подобного рода.

Строка меню под заголовком окна (активизируется клавишей *F10* при ее отсутствии, см. рис. 1,2) состоит из 10 пунктов, которые, в свою очередь, разворачиваются в спускающиеся подменю:

File – позволяет выполнять все основные операции с файлами: создавать новые, загружать имеющиеся, сохранять созданные и отредактированные файлы, выводить на принтер содержимое этих файлов, заканчивать работу с ИИО и так далее.

Edit – дает возможность выполнять основные операции редактирования текста.

Search – позволяет осуществлять поиск фрагментов текста и при необходимости производить замену найденного фрагмента новым.

Run – позволяет запускать программу, находящуюся в рабочей зоне, а так же при необходимости пошагово выполнять данную программу или ее часть. Если были внесены изменения в программу, то при запуске она автоматически заново компилируется.

Compile – возможно осуществить компиляцию программы, которая находится в рабочей зоне, без ее выполнения.

Debug – содержит команды, облегчающие процесс поиска ошибок в программе: расстановка точек останова, визуализация окна отладки, окна регистров, окна выходных результатов и так далее.

Tools – дает возможность выполнять некоторые программы, не выходя из ИИО.

Options – здесь находятся команды, позволяющие установить необходимые для работы параметры компилятора и ИИО.

Window – позволяет выполнять все основные операции с окнами (хотя их удобнее выполнять с помощью мыши): открывать, закрывать, перемещать, изменять размер.

Help – позволяет получить имеющуюся в системе справочную информацию.

Система меню позволяет выполнять практически все команды ИИО и интегрированных программ и выполнена в соответствии со стандартом SAA (Turbo Vision).

Строка состояния, находящаяся в нижней части экрана, в режиме редактирования демонстрирует некоторые из часто используемых операций ИИО и комбинации клавиш для их быстрого вызова, которые позволяют выполнить соответствующие операции, минуя стандартную процедуру их вызова через меню. В некоторых режимах здесь выводятся подсказки или другая справочная информация.

Отладка и выполнение программы

После того, как программа набрана в виде текста в окне редактирования, ее необходимо откомпилировать, устранить синтаксические и семантические ошибки и выполнить, то есть получить конечный результат.

После набора программы ее **рекомендуется сохранить на диске**. Более того, если текст программы достаточно объемен, лучше всего делать и промежуточные записи во избежание потери набранного текста при ошибочных действиях или сбоях компьютера.

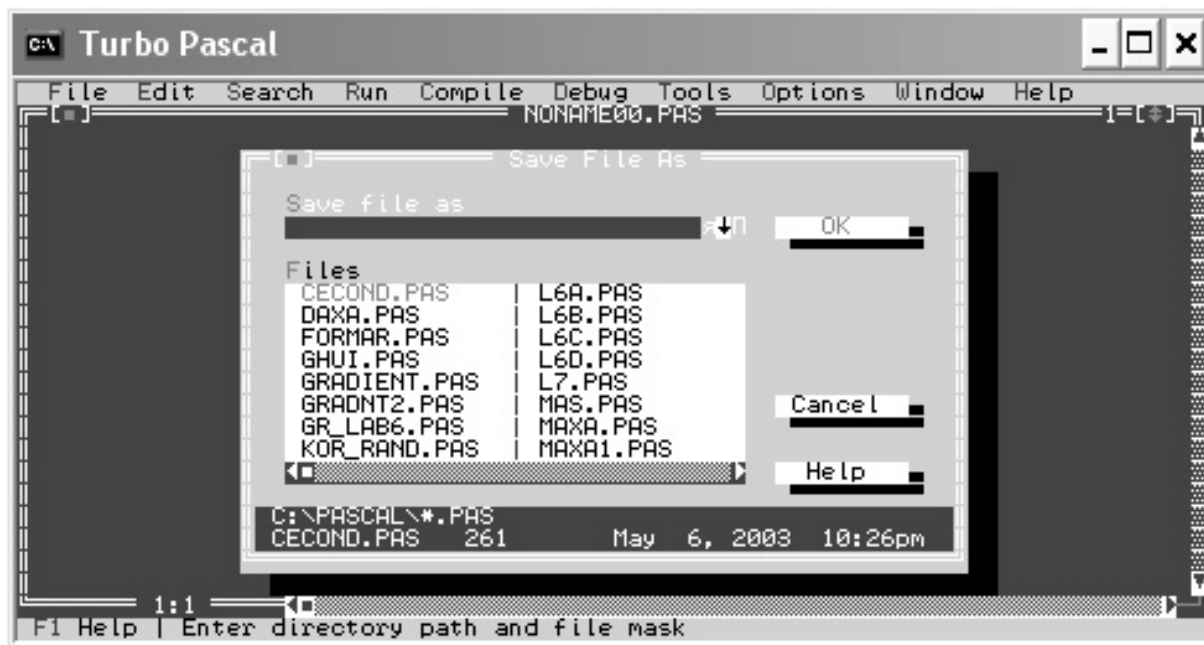


Рис. 3. Сохранение текста программы

Если программа набирается заново, то есть активное окно имеет имя NONAME00.PAS, то при сохранении программы, – нажатии клавиши *F2*, выполнится команда *File/ Save as...* При этом появится диалоговое окно со списком файлов, – программ из текущего раздела, аналогично примеру, приведенному на рис. 3. В области

Save file as набирается имя файла, с которым он будет сохранен на диске. Если не ставить точку и расширение, то имя автоматически будет дополнено расширением **.PAS**. После записи на диск имя в текущем окне редактирования сменится на заданное. После дальнейшего набора программы или ее корректировки при нажатии клавиши **F2** будет выполняться команда *File/Save* (сохранение файла с тем же именем), и никаких дополнительных запросов происходить не будет.

При переходе к новой программе окно с текстом старой программы закрывают (<Alt+F3> или *Window/Close*, хотя это и не обязательно) и **открывают новое окно редактирования** (*File/New*). При необходимости загрузки (чтения с диска) другой, ранее набранной программы, выполняется команда *File/Open...* (**F3**), в появляющемся диалоговом окне, представленном на рис. 4, в области *Files* перемещением маркера выбирается нужный файл и нажимается клавиша *Enter*, эквивалентная кнопке диалогового окна *Open*. Здесь так же можно открывать вложенные папки, их список находится в конце, или переходить к папкам более высокого уровня, выбрав строку «..\».

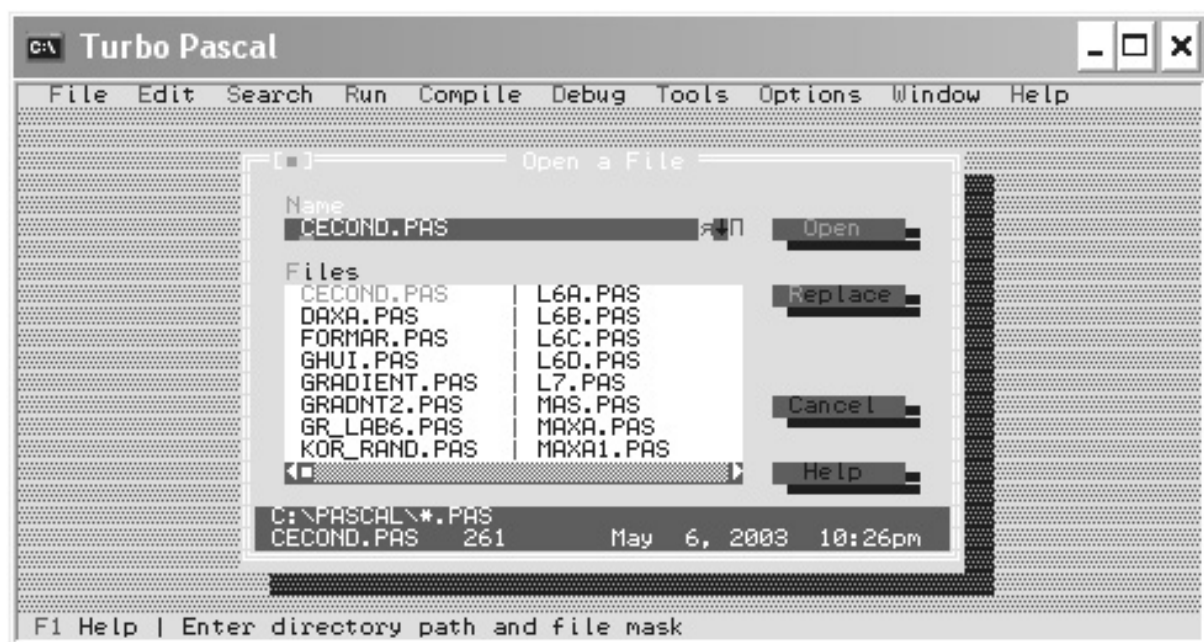


Рис. 4. Загрузка текста программы

После ввода всей программы (или ее загрузки) ее можно либо откомпилировать с целью устранения синтаксических ошибок (<Alt+F9>), либо сразу выполнить (<Ctrl+F9>). В последнем случае все равно произойдет предварительная компиляция, и если присутствуют синтаксические ошибки, программа выполняться не будет.

Все **ошибки программ** делятся на два больших класса: ошибки компиляции и ошибки выполнения. О первом типе ошибок сообщает компилятор до запуска программы на выполнение с указанием типа ошибки и предполагаемого ее места. К сожалению, ошибка может быть совсем не там, где стоит курсор; его положение – это фактически то место, где компилятор «осознает» ошибку. Например, если имеется лишнее слово **BEGIN** в программе, то компилятор не поймет этого до тех пор, пока не дойдет до последней строки программы (пары служебных слов **BEGIN... END** должны быть сбалансированы).

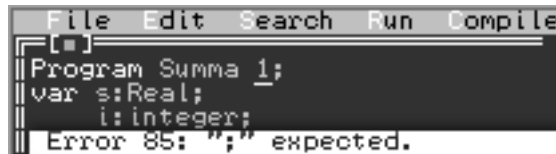
На начальном этапе программирования большинство синтаксических ошибок происходит из-за невнимательности набора программы. Даже в первой строке часто делаются ошибки, как представлено на рис. 5. После компиляции курсор установлен в самой первой позиции (на рисунке буква **P** подчеркнута).



Рис. 5. Пример 1 синтаксической ошибки

В данном примере **пояснение причины ошибки, а это красная строка с текстом «Error 36: BEGIN expected»** (ожидается служебное слово **Begin**), не имеет особого смысла. Здесь служебное слово **Program** пишется с одним **m**, поэтому первая строка не воспринята как начало текста программы. В Турбо Паскале заголовок программы (первая строка) может отсутствовать, так же как и все разделы описаний. Но раздел операторов, начинающихся со слова **Begin**, должен быть обязательно.

Более осмысленное толкование причины ошибки происходит в случае, представленном на рис. 6 во фрагменте окна.

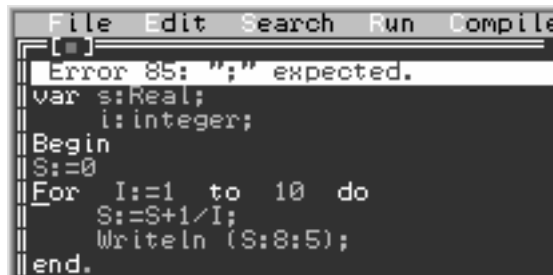


```
File Edit Search Run Compile
Program Summa 1;
var s:Real;
    i:integer;
Error 85: ";" expected.
```

Рис. 6. Пример 2 синтаксической ошибки

Здесь сообщение причины ошибки, «**Error 85: ";" expected**»: перед единицей (где установлен курсор) ожидается точка с запятой. Причина ошибки заключается в том, что имя программы, как и имена переменных, не должно включать пробелы, поэтому за разделителем, – пробелом, – должна идти следующая конструкция языка, отделяемая от заголовка точкой с запятой.

Довольно часто курсор устанавливается в строке, следующей за ошибочной, пример которой представлен на рис. 7 во фрагменте окна.



```
File Edit Search Run Compile
Error 85: ";" expected.
var s:Real;
    i:integer;
Begin
S:=0
For I:=1 to 10 do
    S:=S+1/I;
    Writeln (S:8:5);
end.
```

Рис. 7. Пример 3 синтаксической ошибки

Точка с запятой должна стоять перед **F** (на этой букве установлен курсор), то есть в конце предыдущей строки.

В любом случае при непонимании ошибки следует обратиться к синтаксису отмеченной курсором конструкции языка Турбо Паскаль (оператору, описанию, структуре), либо к предыдущей.

Ошибки выполнения появляются после компиляции программы и запуска ее на выполнение, и на экране с текстом программы выдается сообщение вида

Run-time error < *nnn* > at < *xxxx:yyyy* >

где *nnn* — номер ошибки выполнения, *xxxx:yyyy* — адрес ошибки выполнения.

Ошибки выполнения искать труднее, чем синтаксические. Это и ошибки на уровне операционной системы **DOS**, и ошибки ввода-вывода, и ошибки алгоритма, и другие. Но, хотя ошибок данного класса довольно много, в данном курсе контрольных работ чаще всего появляются только две фатальные ошибки (приводящие к аварийному завершению программы):

200 Division by zero (Деление на нуль)

205 Floating point overflow (Переполнение при операции с плавающей точкой)

Вообще говоря, комментарии к этим ошибкам не требуют дополнительных пояснений, хотя они могут случаться не только при недопустимых делениях и вышедших из под контроля циклах операций умножения, но и при недопустимых аргументах математических функций: отрицательных значениях для логарифмов и квадратных корней, и других.

Все же наибольшую трудность вызывают ситуации, когда никаких сообщений не появляется, а компьютер просто «зависает», или выдает результат, но неверный. При этом начинающий пользователь склонен обвинять компьютер, хотя практически всегда виноват он сам. Эти ошибки возникают либо при неверно составленном алгоритме, а соответственно, и программе, либо при выходе за границы неконтролируемых величин.

Для обнаружения выходов за допустимые границы и переполнений можно использовать две директивы компилятора. По умолчанию они выключены, так как замедляют работу программы и увеличивают ее размер, а используются только при отладке.

Проверка границ. `{SR+}`, эквивалент меню **Options / Compiler**, опция **Runtime errors / Range checking**.

Данная директива приводит в действие генерацию кода проверки границ. Все выражения с индексированными строками и массивы проверяются на предмет нахождения их внутри указанных границ, а все операторы присваивания значений скалярным величинам и переменным поддиапазонов также проверяются на нахождение в заданных границах. Если обнаруживается нарушение диапазона, программа завершает свою работу, выводя сообщение об ошибке выполнения при проверке границ:

201 Range check error

Проверка переполнения при математических операциях. `{Q+}`, эквивалент меню **Options / Compiler**, опция **Runtime errors / Overflow checking**.

Проверяется результат операций над целочисленными данными. Программа завершает работу аналогично:

215 Arithmetic overflow error

ИИО Турбо Паскаля может работать с несколькими окнами, просмотреть список открытых окон можно командой *Window/List...* В окне редактирования набираются, просматриваются и редактируются программы, отдаются команды ИИО, устанавливаются параметры работы и так далее. Но при запуске программы появляется другое окно, – **окно пользователя**, «User screen». Сюда помещаются результаты работы по программе, и оно видно до тех пор, пока программа не перестанет выполняться. Так как в данном курсе контрольных работ программы весьма просты, то они выполняются практически мгновенно, и снова появляется окно редактирования ИИО. **Для просмотра экрана пользователя (в данных работах это результат вычислений по программе) нажимается комбинация клавиш <Alt+F5>.** Для возврата к основному окну можно нажать любую клавишу.

Если после запуска возникнет **необходимость прервать выполнение программы**, например в случае ее «зацикливания», используют комбинацию клавиш <Ctrl+Break>.

В окне пользователя мы увидим либо сообщение об ошибке выполнения, либо результат (если отсутствует оператор вывода, естественно, результата не будет). Результат может быть и неверным как из-за неправильно составленного алгоритма, так и из-за ошибок в программе, которые формально, с точки зрения компилятора, ошибками не являются. Например, при вычислении суммы по фрагменту программы, представленному на рис. 8.

```
File Edit Search Run Compile
Program Summa1;
var s:Real;
    i:integer;
Begin
S:=0;
For I:=1 to 10 do;
    S:=S+1/I;
    Writeln (S:8:5);
end.
```

Рис. 8. Пример логической ошибки в программе

Для этой программы результат вычислений будет равен **0.10000**. Здесь после служебного слова **do** стоит пустой оператор (точка с запятой), поэтому именно он, то есть «ничто», будет выполняться десять раз. Затем к нулю прибавится одна десятая. Формально программа составлена правильно, но фактически десять раз должен выполняться оператор из строки **S:=S+1/I**.

Конкретные рекомендации в таких случаях являются индивидуальными для каждой программы, существуют только общие методы тестирования. Рассмотрим два самых простейших.

Первый метод самый универсальный и может использоваться с любыми языками программирования, компиляторами и ассемблерами. Заключается он в выводе промежуточных результатов. Внутри цикла, или в «подозрительные» места программы временно вставляются операторы вывода изменяемых переменных. Например, программа (приведен фрагмент)

```
For i:=1 to 15 do
  Begin
    S:=S+1/i;
    i:=i+1
  end;
```

просто «заиклиивается» (если не используются директивы контроля). Но если перед $i:=i+1$ поставить оператор

```
Writeln (S:8:5,i:8);
```

то после запуска программы появятся два столбика бегущих чисел. Если приостановить вывод информации на экран монитора **клавишей** *Pause*, то сразу будет видно, что i изменяется не с шагом 1, а 2. То есть параметр цикла изменяется и в операторе **For**, и в операторе присваивания.

Второй метод привлекает средства отладки ИИО. Предварительно установив программный счетчик на начало программы (<Ctrl+F2>), открывается специальное окно наблюдаемых переменных (**Watch**-окно, клавиши <Ctrl+F7>). В нем набирается имя первой контролируемой переменной, нажимается *Enter*. После этого появляется окно с именем **Watches** и значением этой переменной, в которое можно добавить и другие переменные для контроля их значений. Для пооператорного выполнения программы предназначена «горячая» клавиша *F7*. После каждого нажатия клавиши *F7* выполняется один оператор. Если он производит изменение значения переменной, то это сразу отразится в окне наблюдения.

Таким образом, анализируя изменение значений переменных при пооператорном выполнении программы, легко найти ошибку в алгоритме, и соответственно в программе.

По завершении работы с ИИО, из нее можно выйти по комбинации клавиш <Alt+X>.

СПИСОК ВОПРОСОВ ПО ТЕОРЕТИЧЕСКОЙ ЧАСТИ

Для выполнения контрольных работ необходимо ознакомиться с правилами составления программ на языке Паскаль. Краткий список литературы с описанием языка Паскаль и приемов программирования на этом языке приведен в соответствующем разделе [1-6].

Для освоения теоретической части и подготовке к сдаче экзамена необходимо изучить следующие вопросы. В них для краткости Турбо Паскалем называется расширение языка программирования Паскаль в среде программирования Турбо Паскаль 7.0.

1. Понятие и свойства алгоритма
2. Формы записи алгоритмов, типовые блоки в блок-схемах алгоритмов
3. Основные типы (структуры) алгоритмов
4. Язык Паскаль и среда Турбо-Паскаль, расширение языка, алфавит и служебные слова
5. Структура программы на языке Паскаль, общие правила составления программы
6. Разделы описаний
7. Иерархия типов в языке Паскаль
8. Стандартный целый тип данных и целые типы в Турбо Паскале
9. Стандартный вещественный тип данных и вещественные типы в Турбо Паскале
10. Литерный тип данных
11. Логический тип данных
12. Перечисляемый тип данных, диапазоны (интервальный тип)
13. Выражения, порядок выполнения операций, основные стандартные функции
14. Оператор присваивания, составной и пустой операторы
15. Операторы ввода-вывода при работе со стандартными устройствами, форматный вывод
16. Условный оператор **IF** и безусловный оператор перехода
17. Оператор цикла с параметром **FOR**
18. Оператор цикла с предусловием **WHILE**
19. Оператор цикла с постусловием **REPEAT**
20. Оператор выбора **CASE**
21. Подпрограммы-функции

22. Подпрограммы-процедуры
23. Рекурсия. Директивы для подпрограмм
24. Структурированные типы данных: Массивы и строки
25. Структурированные типы данных: Множества
26. Структурированные типы данных: Записи
27. Файлы, основные понятия и приемы работы
28. Обработка ошибок ввода-вывода
29. Текстовые файлы, стандартные текстовые файлы
30. Понятие динамических переменных, указатели
31. Приемы работы с динамическими переменными
32. Встроенный язык Ассемблер
33. Доступ к памяти и портам ввода-вывода
34. Работа по прерываниям
35. Понятие модуля в Турбо Паскале
36. Использование модулей
37. Стандартные модули, их назначение

В данный курс не входят вопросы объектно-ориентированного программирования и применение конкретных процедур и функций стандартных модулей (кроме модуля **System**), то есть то, что относится только к расширению языка – Турбо Паскалю.

ПОРЯДОК ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Курсовая работа состоит из двух частей, каждая из которых содержит два задания. По каждому заданию составляется алгоритм и программа (для задания 1.2 две программы), которые помещаются в отчет.

Отчет оформляется в ученических тетрадях **12** листов или на стандартных листах формата **A4**, которые затем скрепляются.

Отчет должен состоять из титульного листа и описания заданий..

Титульный лист оформляется в соответствии с общими требованиями соответствующего факультета по оформлению титульных листов контрольных работ. В любом случае на первом листе (для тетради – обложке) должны находиться сведения по названию предмета, номеру, названию и варианту контрольной работы, ФИО учащегося, его шифру (номеру зачетной книжки).

Номер варианта определяется в соответствии с последними двумя цифрами шифра (номера зачетной книжки) по табл. 1.

Таблица 1. Соответствие шифра и номера варианта.

Номер варианта	Шифр	Номер варианта	Шифр	Номер варианта	Шифр
1	01,31,61,91	11	11,41,71	21	21,51,81
2	02,32,62,92	12	12,42,72	22	22,52,82
3	03,33,63,93	13	13,43,73	23	23,53,83
4	04,34,64,94	14	14,44,74	24	24,54,84
5	05,35,65,95	15	15,45,75	25	25,55,85
6	06,36,66,96	16	16,46,76	26	26,56,86
7	07,37,67,97	17	17,47,77	27	27,57,87
8	08,38,68,98	18	18,48,78	28	28,58,88
9	09,39,69,99	19	19,49,79	29	29,59,89
10	10,40,70,00	20	20,50,80	30	30,60,90

Каждое задание должно быть представлено четырьмя частями, выполненными рукописно или в виде распечатки электронного

1. Вариант задания. В кратком виде приводится задание на выполнение по конкретному варианту для возможности контроля работы, не обращаясь к данным указаниям. **Например:**

Вычислить $\sum_{i=1}^{\infty} \frac{1}{i^2 + x}$ при $x = 4,376$ и заданной точности 10^{-4} .

2. Схема алгоритма. Приводится рисунок с указанием фигур блоков с конкретным внутренним содержанием согласно варианту задания. Оформление схемы должно соответствовать ГОСТ 19.701-90 [7].

3. Текст программы. Размер программы должен быть минимальным. Не следует использовать различные украшения при выводе программы: использовать графический режим, модуль CRT для очистки экрана, перемещения курсора, ожидания ввода и другие стандартные подпрограммы и модули. Если же в программе они присутствуют, то необходимо четко представлять их назначение и принципы работы, а так же отразить в алгоритме.

В задании 1.2 должно быть приведено два варианта текста программы.

4. Результат вычислений. В данных работах это число, кроме задания 1.2, которое получается после выполнения программы в среде Турбо Паскаль. Для получения **правильного ответа** программа должна быть набрана на персональном компьютере и в ней устранены все синтаксические и логические ошибки.

Именно на этом этапе выполнения работы используется интегрированная среда разработки Турбо Паскаль, основные приемы работы с которой описаны в начале методических указаний.

Этот этап при наличии компьютера можно выполнить самостоятельно, либо на практических занятиях во время сессии. В первом случае можно продемонстрировать умение работать в Турбо-среде, имея программу в электронном виде. Во втором случае контрольная работа сдается без выполненного четвертого пункта: он заполняется во время практических занятий на сессии.

В любом случае **должен быть представлен отчет в бумажной форме.**

ЧАСТЬ 1.

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЛИНЕЙНЫХ И ВЕТВЯЩИХСЯ АЛГОРИТМОВ

Задание 1.1. Программирование формул

Целью работы является освоение программирования **алгоритмов с линейной структурой**, когда решение задачи является результатом выполнения цепи вычислений, в которой очередные вычислительные действия используют в качестве исходных данных результаты вычислений на предыдущих этапах.

Задание является заголовком столбцов таблицы вариантов заданий и формулами в соответствии с вариантом задания.

Действия по вычислениям промежуточных и окончательных результатов описываются операторами присваивания. Необходимо следить, чтобы порядок расположения операторов присваивания в программе от ее начала к концу соответствовал логической последовательности действий при решении поставленной задачи.

Не следует выражать одни переменные через другие для получения одной формулы из нескольких: в алгоритме и программе необходимо записать столько формул, сколько приводится в задании.

При выборе имен переменных и составлении арифметических выражений необходимо правильно устанавливать тип используемых величин (целые, вещественные и так далее). При использовании в формулах греческого алфавита можно использовать их латинские названия или буквы, сходные по начертанию. Например, символ a можно заменить на **Alfa** или **A**, w на **Omega** или **W**.

К сожалению, в языке Паскаль имеется ограниченное количество математических функций, поэтому, при отсутствии стандартной функции ее выражают через другие, используя функциональные соотношения. Ниже приводятся основные математические функции, отсутствующие в языке Паскаль:

$$\arcsin x = \operatorname{arctg} \frac{x}{\sqrt{1-x^2}} [x^2 < 1] ;$$
$$\arccos x = \begin{cases} \operatorname{arctg} \frac{\sqrt{1-x^2}}{x} [0 < x \leq 1] \\ p + \operatorname{arctg} \frac{\sqrt{1-x^2}}{x} [-1 \leq x < 0] \end{cases} ;$$

$$\operatorname{arccctg} x = \begin{cases} \operatorname{arctg} \frac{1}{x} [x > 0] \\ p + \operatorname{arctg} \frac{1}{x} [x < 0] \end{cases} ;$$

$$\operatorname{sh} x = \frac{1}{2} (e^x - e^{-x}) ;$$

$$\operatorname{ch} x = \frac{1}{2} (e^x + e^{-x}) ;$$

$$a^x = e^{x \ln a} \quad [a > 0] ;$$

$$\log_a x = \frac{1}{\log_x a} = \frac{\log_b x}{\log_b a} \quad [x > 0] .$$

Более того, любую функцию можно вычислить с помощью четырех арифметических операций итерационными методами или разложением в ряды.

Варианты задания, определенные в соответствии с табл. 1, приведены в табл. 2.

Таблица 2. Варианты заданий 1.1

№ вар.	Вычислить выражение	При заданных значениях
1	$f(t) = \frac{x}{t} + \frac{t}{y} + \frac{y}{x}$	$y = \sqrt{\frac{t^2 + x}{\ln tx}}$; $x = \arccos t + 9,9$; $t = 0,11$
2	$j(t) = x^3 e^{-xy} \frac{\sqrt{0,8xyt}}{\cos x \cdot \ln y}$	$y = \frac{t+4}{\sin tx}$; $x = \sin t \cdot 1,8$; $t = 0,392$
3	$f(y) = e^{-z} \frac{\operatorname{tg} xy + y^z}{z \cdot \cos^2 xy}$	$x = 0,1 \sqrt{zy} \cdot \frac{\sin zy}{y}$; $z = 0,3 \cdot \ln y$; $y = 2,52$
4	$b(t) = \frac{w}{v} + \sqrt{\left \frac{\sin tw}{\cos tv} \right }$	$w = \operatorname{tg}^2 t$; $v = \operatorname{ctg}^2 t + 0,5$; $t = 0,15$
5	$z(g) = \frac{g^2 + t^2}{\sin vt} + \frac{v^2 + g^2}{\cos gt}$	$v = \sqrt{g^2 + t^2}$; $t = 0,5 \operatorname{tg}^2 g$; $g = 9,8$
6	$a(p) = 10^{-p} \sqrt{\sin wp + \cos gp}$	$w = 2g p^2$; $g = \arcsin^3 p$; $p = 0,15$
7	$w(t) = \frac{e^{-abt} \cdot \operatorname{tg}^3 bt}{1 + \cos at}$	$a = \ln bt + 2$; $b = \sqrt{\frac{t+1}{t}}$; $t = 1,11$
8	$j(w) = \arcsin \left(w \frac{pq}{p+q+1} \right)$	$p = \lg wq$; $q = 1 + \frac{1}{w^2}$; $w = 0,28$
9	$g(a) = a + \left(\frac{\sin abc}{a+b+c} \right)^{0,48}$	$b = \frac{0,34}{a+c}$; $c = 0,9 + a^2$; $a = 1,98$
10	$Q(g) = \lg abg + 10^{-abg}$	$a = \sqrt{b^2 + g^2}$; $b = 0,8g^3$; $g = 1,33$
11	$g(x) = e^{-zyx} \frac{x^2 + y^2 + z^2}{\sqrt{xyz}}$	$z = \ln \frac{x}{y}$; $y = 0,55 \sqrt{x}$; $x = 1,19$
12	$a(t) = \sin \frac{wr}{t} \sqrt{\frac{w+r+t}{wrt}}$	$w = r \sqrt{\frac{1}{t}}$; $r = 10^{-t}$; $t = 1,58$

1	2	3
13	$b(x) = \frac{\cos^2 \frac{a}{bx} + 2}{a^2 + \sin bx} + \frac{bx}{a}$	$a = \ln \sqrt{\frac{b}{x}}; b = 2e^{-x}; x = 2,35$
14	$g(y) = \operatorname{tg} \frac{y}{b} + \operatorname{tg} \frac{b}{c} + \operatorname{tg} \frac{c}{y}$	$b = \frac{\sin c}{\cos y}; c = 4 \cdot \sqrt{\frac{0,8}{y}}; y = 2,25$
15	$p(z) = \frac{\ln c + \operatorname{lg} bz}{b^2 + c^2 + z^2}$	$c = \sqrt{\frac{\sin z}{\cos b}}; b = \arcsin z; z = 0,82$
16	$S(g) = \operatorname{lg} \frac{w^3}{dg} + \operatorname{lg} \sqrt{\frac{wd}{g^3}}$	$d = w^2 + 0,7g; w = 10^{-g} \cdot g^2; g = 1,83$
17	$j(w) = \frac{\operatorname{tg} wx}{\sin tx + 1} \cdot \frac{w}{t}$	$x = \sqrt{\ln wt + 1}; t = \frac{1+w}{1-w}; w = 0,87$
18	$g(p) = \operatorname{lg} pq + \ln \sqrt{pg}$	$q = e^{-p+1} + e^{-g+1}; g = \operatorname{tg} \sqrt{p}; p = 0,65$
19	$g(s) = \frac{s^2 + a^2}{s^2 - b^2} \cdot ab$	$a = 0,7 \sqrt{b^2 + s^2}; b = 10^{s-1}; s = 1,35$
20	$t(h) = e^{-hp} + 10^{-hg}$	$p = \frac{\sin^2 hg}{h^2 + g^2}; g = 0,8 \arcsin \frac{1}{h}; h = 2,4$
21	$u(w) = \frac{y^2 - w^2}{\cos yw} + \frac{x^2 - w^2}{\sin xw}$	$x = \frac{\sqrt{y+w}}{\operatorname{tg} wy}; y = w^2 - 2,5; w = 2,1$
22	$z(a) = \sqrt{a+b+c} \cdot e^{-abc}$	$b = \ln^2(c+1) - \operatorname{lg}^2(a+1); c = \sqrt{a};$ $a = 1,2$
23	$N(b) = \frac{\arcsin bp}{\arccos bg} + \sqrt{p+g}$	$p = 1 - e^{-bg}; g = \sqrt{b^2 + 0,1b};$ $b = 0,7$

1	2	3
24	$R(y) = \frac{(x+y+z)^{0,15}}{x^2 + y^2 + z^2}$	$x = \frac{\operatorname{tg} yz}{\sin y + \cos z}; z = 0,8 + \sqrt{0,3y};$ $y = 0,45$
25	$w(g) = \frac{g^2 p + n^2 + p^2 n}{\sin g p + \cos n g + \sin p n}$	$p = \sqrt{n^3 g + g^3 n}; n = 0,8e^{-2g};$ $g = 0,63$
26	$r(c) = \arcsin \sqrt{\frac{abc + a^2 + b^2}{1 + abc + a^2 + b^2}}$	$a = \frac{\sqrt{b^3 + c^3}}{b + c + 1}; b = \operatorname{tg} \frac{1}{c + 1}; c = 2,4$
27	$p(t) = \frac{\sqrt{xyt} + 2xy}{\sqrt{x + y + t} + 2yt}$	$x = \frac{\sin(y+t)}{\cos(y-t)}; y = \ln 2t; t = 1,2$
28	$c(m) = \ln \frac{m^2 + p^2 + c^2}{mpc + 1}$	$p = \frac{2cm + 1}{c^3 + m^3 + 1}; c = m + \sqrt{m};$ $m = 4,7$
29	$b(d) = \frac{e^{-kd}}{kdz + 2z^3}$	$k = \frac{\sin d}{\cos z}; z = \operatorname{tg} \frac{d}{d + 1}; d = 0,15$
30	$x(l) = \ln \frac{l^2 + fg}{l^2 + f + g + 1} +$ $+ \operatorname{lg} \frac{f^2 + gl}{g^2 + l + f + 1}$	$f = \sqrt{\frac{2g^2 + 3l^2}{gl + 1}}; g = 1,3 \sin \frac{1}{l};$ $l = 2,7$

Задание 1.2. Ветвящиеся алгоритмы

Логические выражения используются не только для решения задач булевой алгебры, но и для ветвления программы в логических и циклических операторах. Причем последний вариант использования логических выражений применяется наиболее часто.

Логические выражения состоят из логических констант, переменных и отношений, соединенных логическими операциями. В простейших случаях в операторах используют отношения: два выражения, соединенных знаком отношения или сравнения ($<$, $>$, $>=$, $<=$, $=$, $<>$), например $I > 20$. Но иногда возникают условия, требующие использования более сложных логических выражений.

Пример задания. На плоскости задана фигура (например, рис. 9, усеченный круг, в заданиях область фигуры заштрихована). Вводится точка с координатами X, Y . Определить, принадлежит введенная точка фигуре или нет. В результате анализа должно выводиться: «Введенная точка принадлежит фигуре» или «Введенная точка фигуре не принадлежит». Считать, что точка на границе фигуре принадлежит.

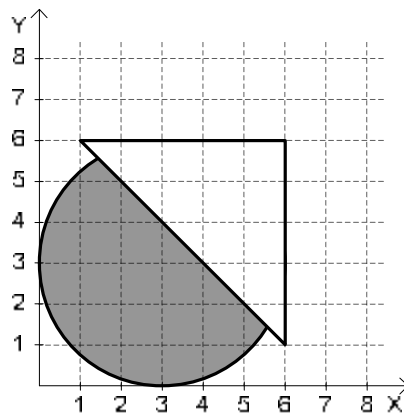


Рис. 9. Пример фигуры

Для определения вхождения точки в круг можно использовать формулу окружности

$$(X - X_0)^2 + (Y - Y_0)^2 = R^2.$$

Соответственно изменив знак $=$ на \leq получим условие вхождения точки в круг с координатами центра $(3, 3)$ и радиусом 3 :

$$(X - 3)^2 + (Y - 3)^2 \leq 9.$$

Кроме этого область, занятая треугольником, так же не входит в закрашенную область, то есть полуплоскость над прямой

$Y = -X + 7$ фигуре не принадлежит. Условия нахождения точки внутри круга и под прямой должны выполняться одновременно. Для этого необходимо использовать логическую операцию *AND*.

Таким образом логическое выражение

$$(X-3)^2 + (Y-3)^2 \leq 9 \quad \text{AND} \quad Y \leq -X + 7$$

примет истинное значение, если точка входит в закрашенную область, иначе ложное. Тогда в логическом операторе по прямой ветви **Then** выводится «Введенная точка принадлежит фигуре», а по ветви **Else** – «Введенная точка фигуре не принадлежит».

Но можно и поменять ветви местами, тогда при вхождении точки в фигуру логическое выражение должно принимать ложное значение. Тривиальный вариант: поставить перед предыдущим выражением знак отрицания *NOT*. Но более наглядным решением будет составление выражения с условием невхождения точки в фигуру. Здесь должно выполняться хотя бы одно из условий: точка не входит в круг или точка лежит над прямой, соответственно, логическое выражение примет вид:

$$(X-3)^2 + (Y-3)^2 > 9 \quad \text{OR} \quad Y > -X + 7$$

При выполнении контрольной работы составить **два варианта программы** (без использования операции *NOT*) для фигуры, соответствующей варианту задания.

Требование данной работы: описать всю фигуру одним логическим выражением.

Алгоритм этой задачи представляет типичную ветвящуюся структуру с одним блоком ввода, одним блоком решения (анализа вхождения точки) и двумя блоками вывода. Так как размеры блоков должны быть одинаковыми (ромб блока решения не должен превышать блоков ввода-вывода), то при необходимости можно использовать фигуру комментария, представленную на рис. 10.

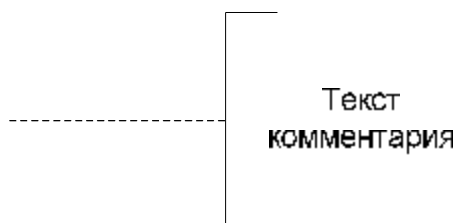
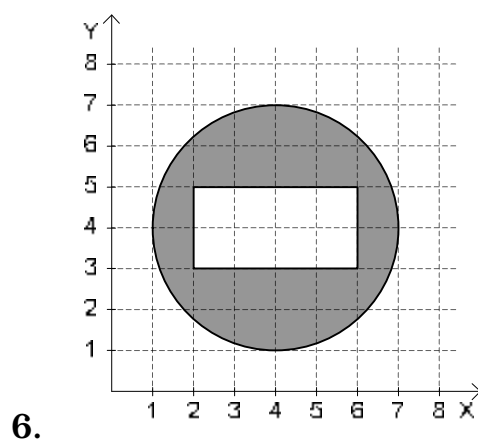
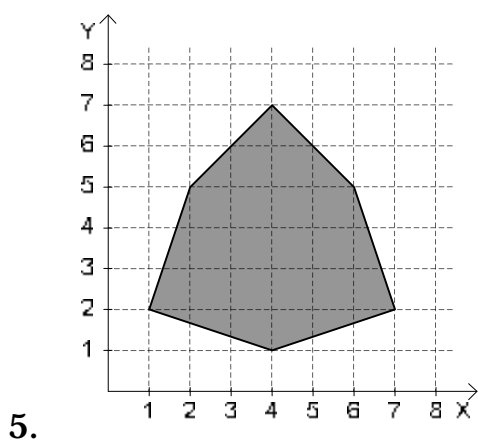
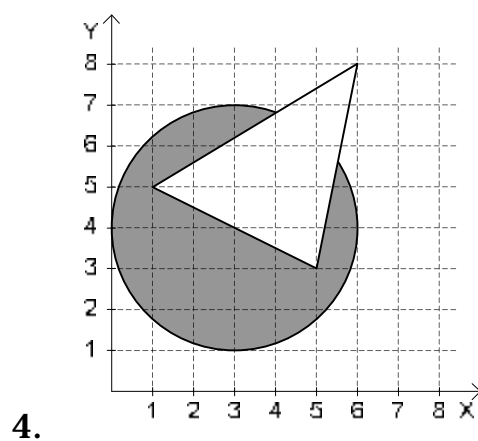
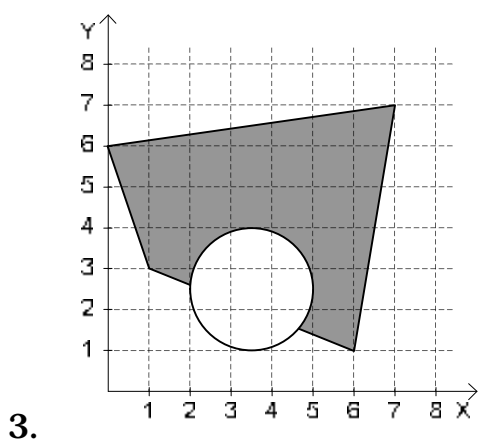
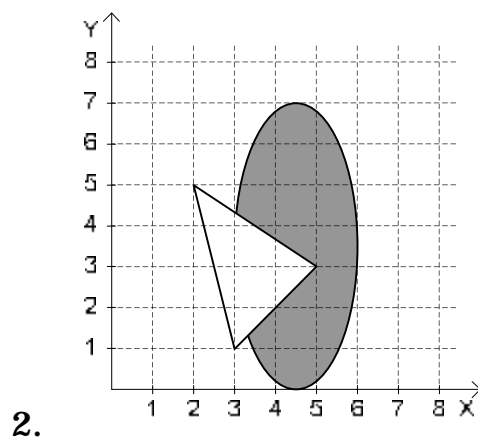
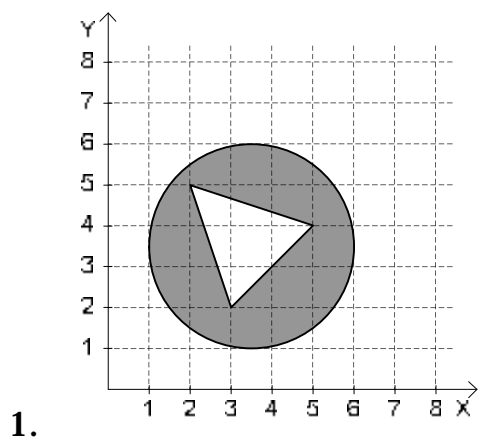
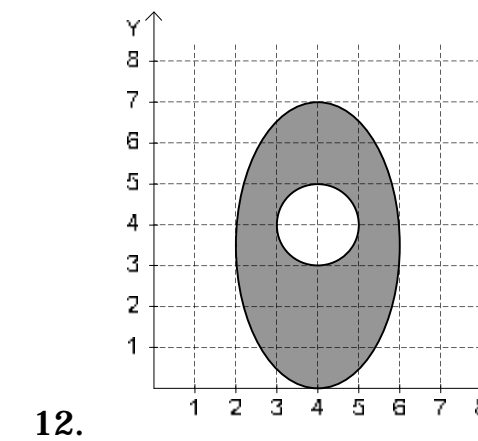
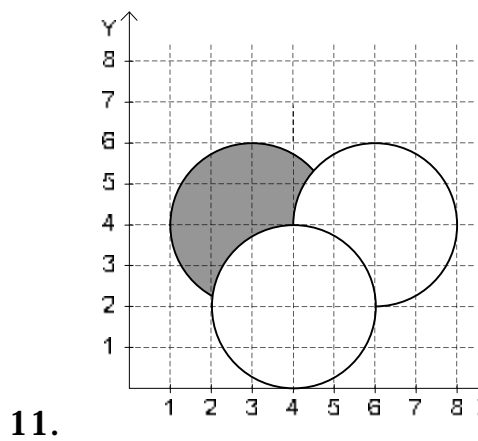
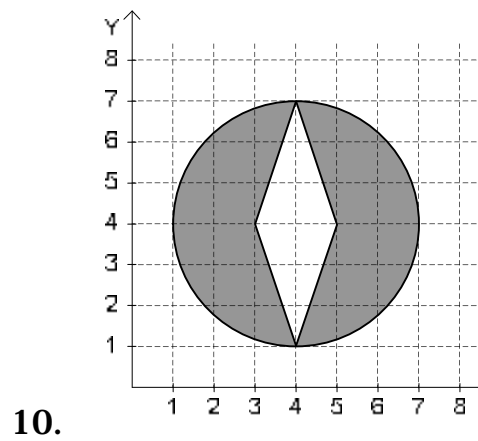
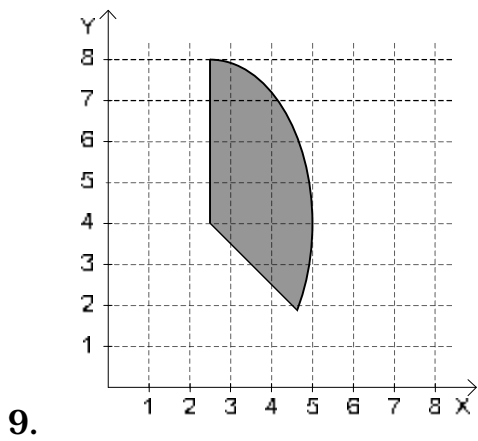
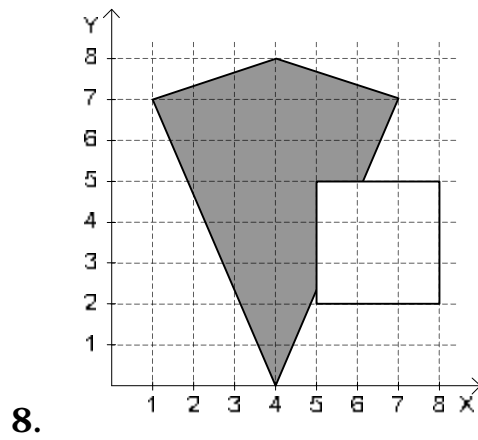
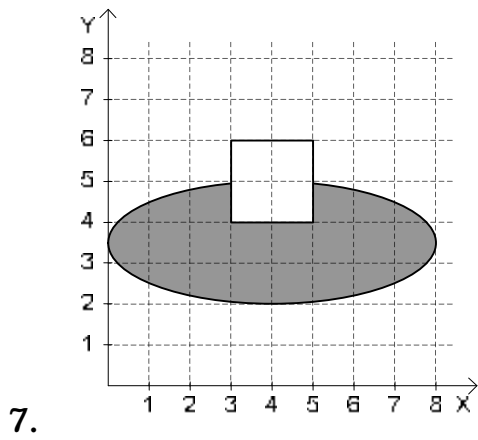
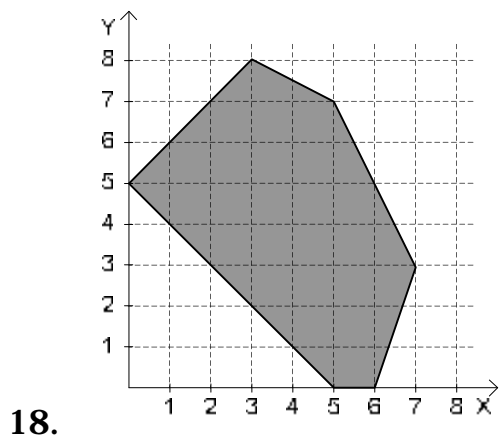
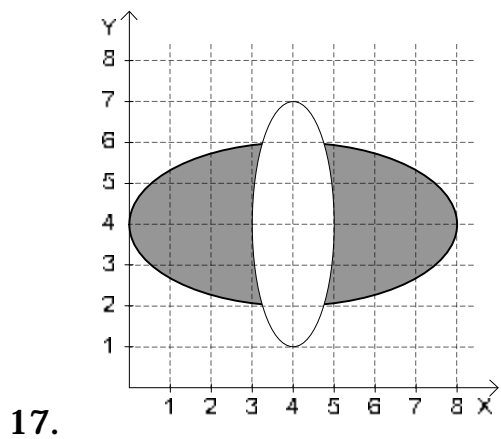
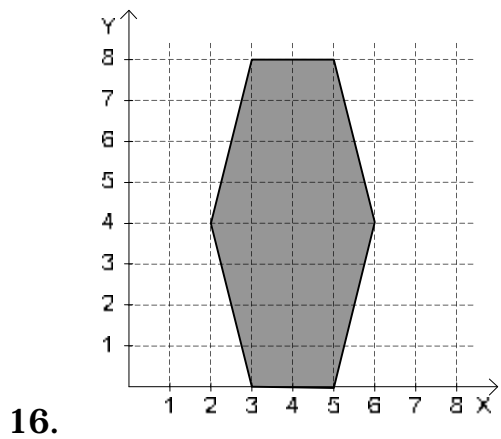
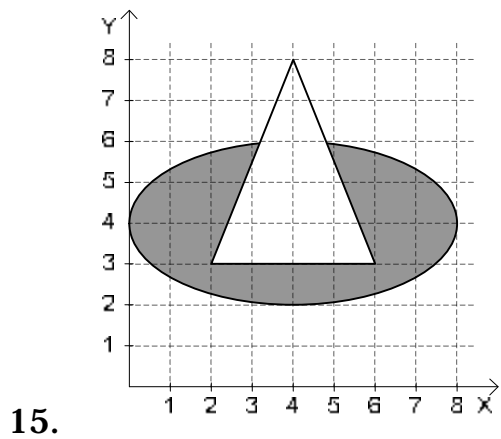
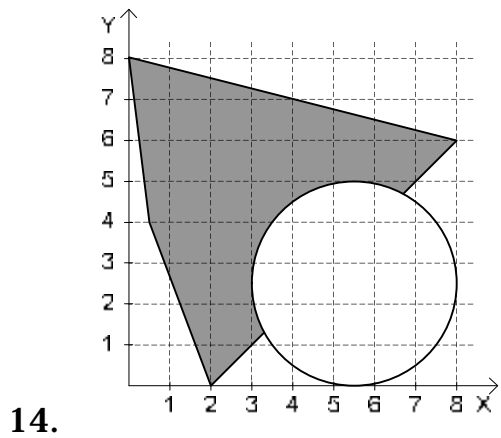
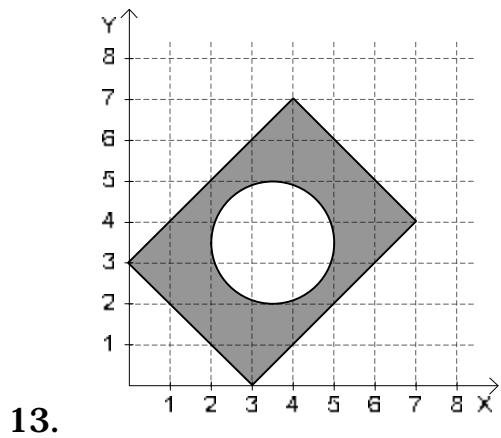


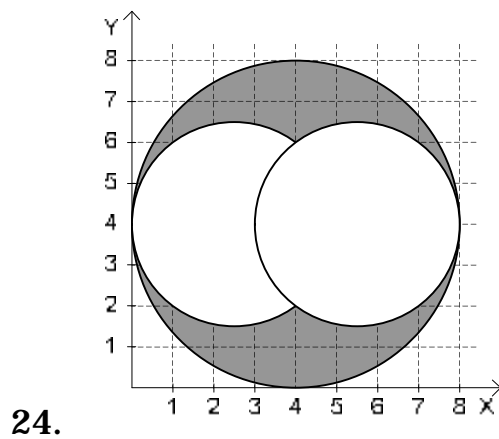
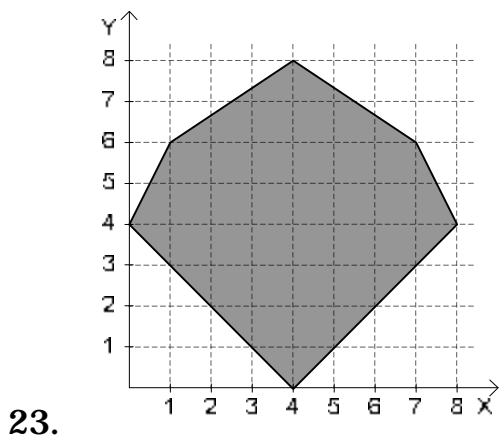
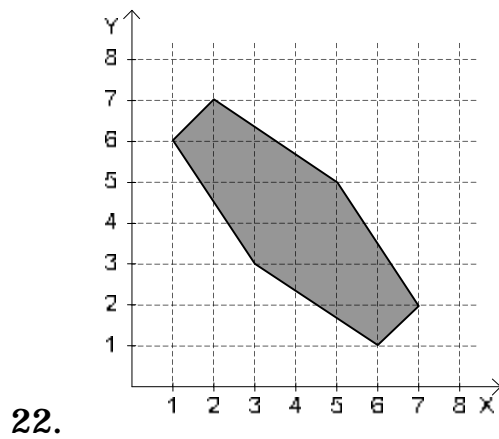
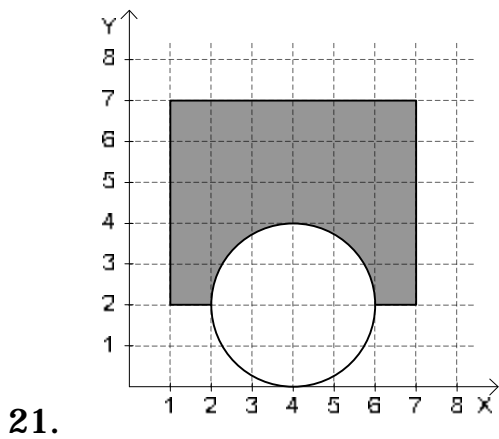
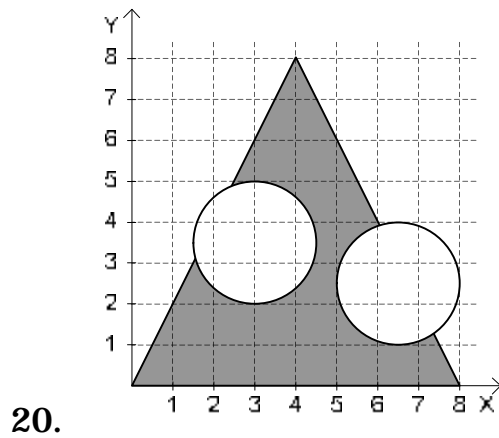
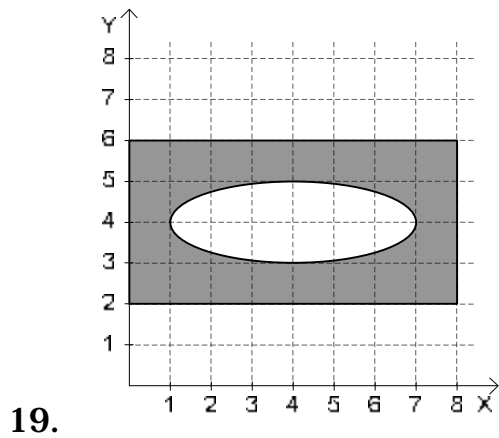
Рис. 10. Комментарий в блок-схемах алгоритмов

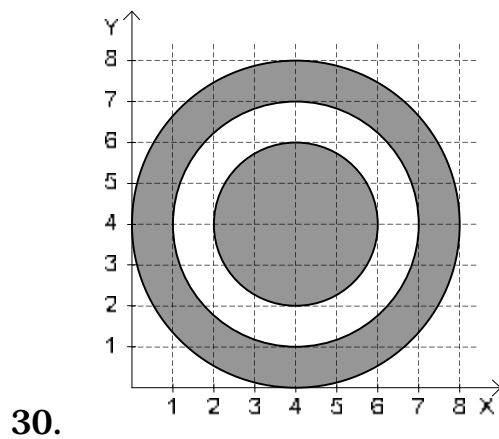
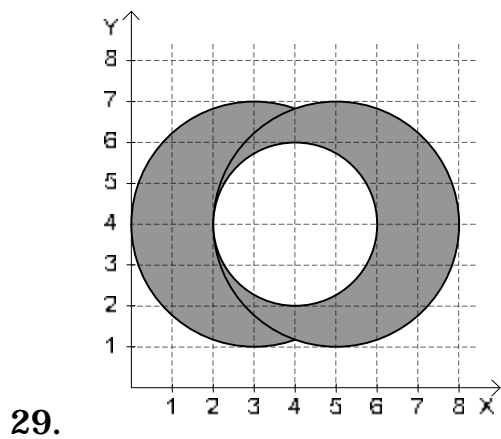
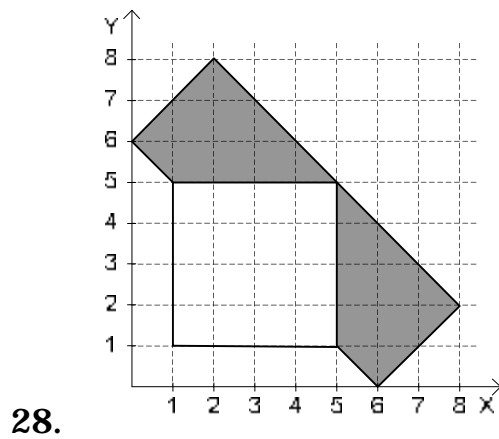
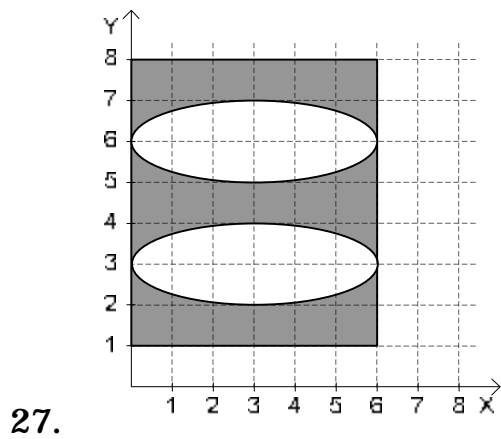
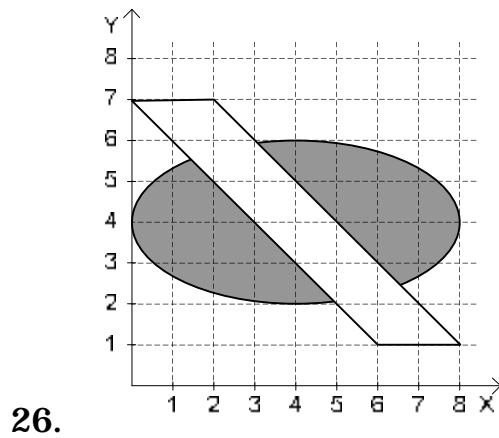
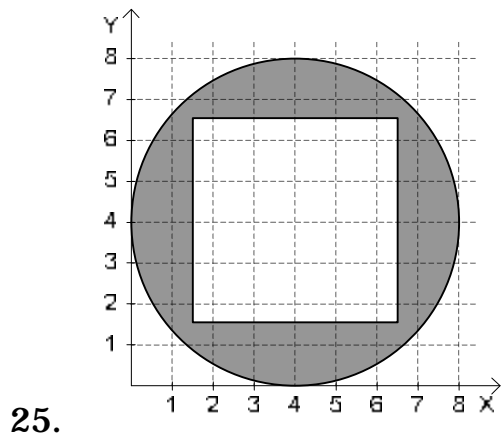
Варианты заданий











ЧАСТЬ 2.

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Задание 2.1. Циклы с известным числом повторений

Целью работы является освоение программирования алгоритмов с циклической структурой, когда какой-либо участок программы выполняется определенное количество раз.

Типичный пример циклического процесса – вычисление конечных сумм. При определении сумм многократно вычисляется выражение, стоящее под знаком суммы и складывается с предыдущей частичной суммой. Вычисления производятся до тех пор, пока не будут сложены выражения под знаком суммы для всех значений изменяющейся переменной.

Пример задания. Составить программу, вычисляющую значение суммы

$$S = \sum_{i=1}^{10} \frac{i}{i+1} .$$

Прежде чем вычислять выражение под знаком суммы и очередную частичную сумму, необходимо определить начальное значение параметра цикла (в данном случае i , которое изменяется от 1 до 10 с шагом 1, то есть i будет принимать последовательно значения 1, 2, 3, 4, ..., 9, 10), и начальную частичную сумму S . Так как вычисления еще не производились, то $S = 0$.

Затем вычисляется выражение под знаком суммы для $i = 1$, затем $i = 2, 3, \dots$ до 10 и каждый раз складывается с предыдущей частичной суммой S_{i-1} . При этом получается новая частичная сумма S_i . После этого i увеличивается на единицу и проверяется, не стало ли $i > 10$. Если еще меньше или равно 10, то вычисляется новая частичная сумма, в противном случае вычисление суммы будет закончено, и это значение выводится на печать.

Воспользуемся стандартной схемой циклического процесса, представленной на рис. 11.

Блок 2 – блок подготовки к вычислению суммы, в котором задаются начальные значения параметра цикла и частичной суммы.

Блок 3 – блок проверки окончания цикла. Необходимо проверить, стало ли i больше 10. Если стало, то цикл закончен, следующим должен выполняться блок печати. Если нет, то вычисление частичной суммы продолжается дальше, то есть выполняются блоки 4 и 5.

В блоке 4 производится вычисление выражения, стоящего под знаком суммы и сложение с предыдущей частичной суммой S_{i-1} . В итоге получается новая частичная сумма S_i .

В блоке 5 происходит изменение параметра цикла (увеличение i на 1). Это блок подготовки к новому циклу.

Проверка может осуществляться условным оператором **IF**, но для организации циклов в языке Паскаль специально предусмотрены три оператора цикла. Если количество повторений заранее известно и параметр является целым числом, то целесообразно использовать оператор **FOR**, включающий в себя блоки 2, 3, 5. В этом случае в алгоритме можно применить блок «Модификация».

Алгоритм для примера с использованием оператора **FOR** приведен на рис. 12. Варианты заданий – в табл. 3.

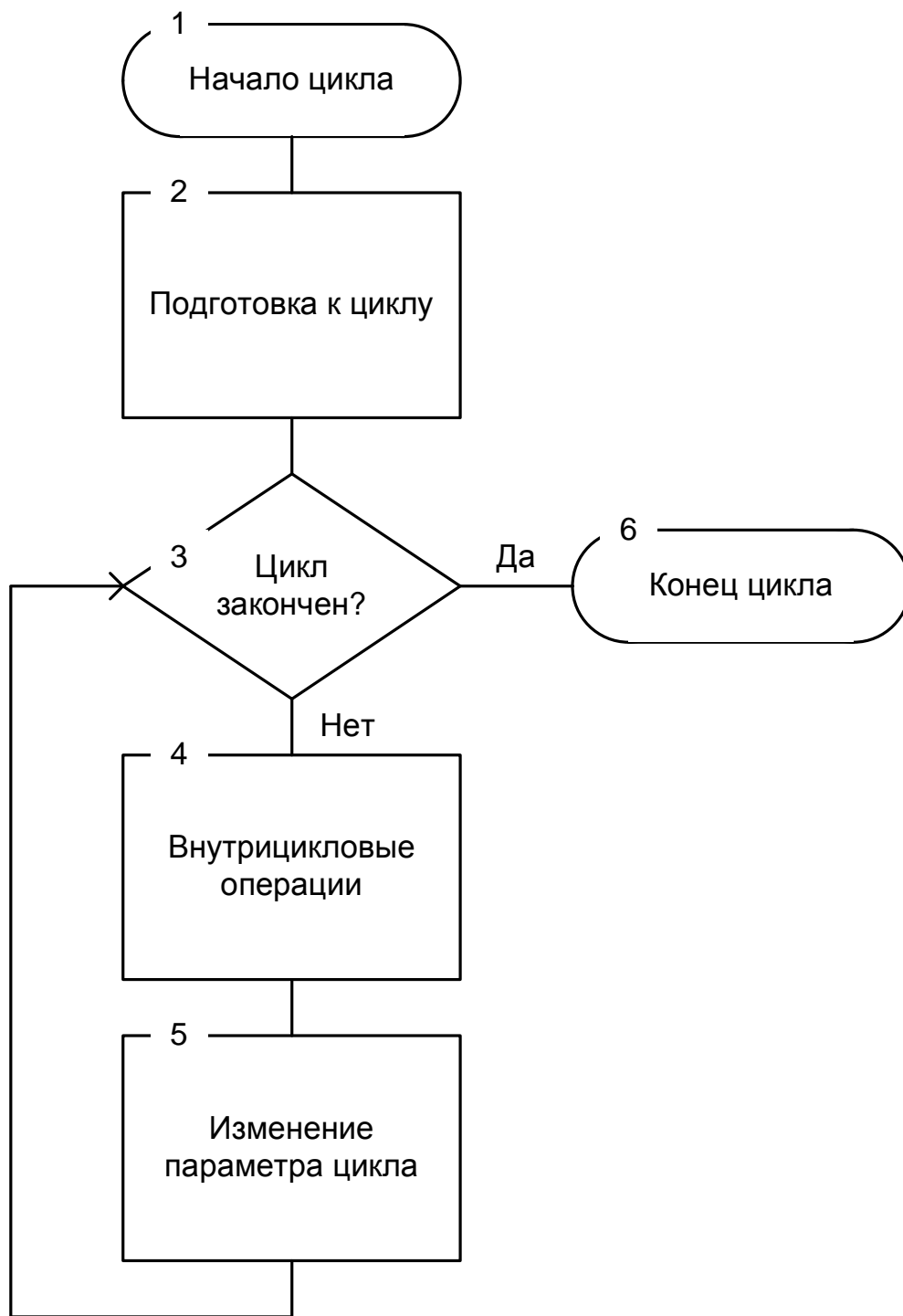


Рис. 11. Блок-схема алгоритма циклического процесса

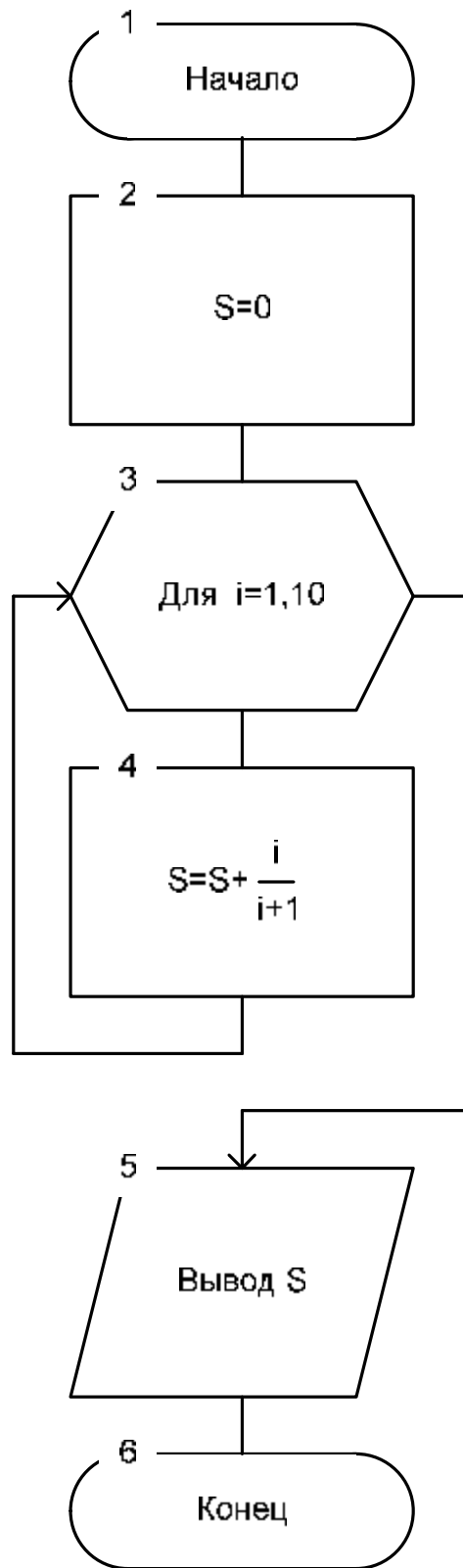


Рис. 12. Блок-схема алгоритма примера.

Таблица 3. Варианты заданий 2.1

№ вар.	Вычислить сумму	№ вар.	Вычислить сумму
1	$\sum_{i=1}^{15} \sqrt{i^2 + 5}$	2	$\sum_{j=1}^{20} \frac{\operatorname{tg} j}{\operatorname{tg} (0,5j + 1)}$
3	$\sum_{k=1}^{25} \ln \frac{k}{k^2 + 1}$	4	$\sum_{l=1}^{30} \frac{e^{0,3l}}{\sin (l + 2)}$
5	$\sum_{m=1}^{15} \operatorname{tg} (e^{1/m} + 1)$	6	$\sum_{n=1}^{20} \frac{\sin (n - 2)}{\operatorname{tg} (n + 2)}$
7	$\sum_{i=1}^{25} \arccos 0,35 \frac{i}{i + 1}$	8	$\sum_{j=1}^{30} \frac{\sqrt{j^3 + j + 3}}{\arcsin 1/(j + 1)}$
9	$\sum_{k=1}^{15} e^{0,1k}$	10	$\sum_{l=1}^{20} \frac{\sqrt{1/l + l}}{\sqrt{3l - 1}}$
11	$\sum_{m=1}^{25} \ln m + e^{-m}$	12	$\sum_{n=1}^{30} \frac{e^{1/n}}{\sqrt{n^2 + 1}}$
13	$\sum_{i=1}^{35} 10^{0,1i} - \arccos \frac{1}{i + 1}$	14	$\sum_{j=1}^{15} \frac{\lg (j + 1)}{\operatorname{tg} (j + 1)}$
15	$\sum_{k=1}^{20} e^{\operatorname{tg} 1/k}$	16	$\sum_{l=1}^{25} \frac{\arccos 0,02l}{\sqrt{2l - 1}}$
17	$\sum_{m=1}^{30} \arcsin \frac{1}{m + 2} \cdot \arccos \frac{1}{m + 1}$	18	$\sum_{n=1}^{25} \frac{\ln n}{e^{-0,2n}}$
19	$\sum_{i=1}^{20} \cos (e^{0,1i} + i)$	20	$\sum_{j=1}^{25} \frac{\lg (j + 2)}{\cos (j + 1)}$
21	$\sum_{k=1}^{30} k^{0,3} + e^{0,3k}$	22	$\sum_{l=1}^{15} \frac{\operatorname{tg} (l - 3)}{\ln (l + 3)}$

Продолжение табл. 3

1	2	3	4
23	$\sum_{m=1}^{20} \sin m \cdot \operatorname{tg} \frac{1}{m}$	24	$\sum_{n=1}^{25} \frac{e^{-0,1n}}{\operatorname{tg}(n-1,5)}$
25	$\sum_{i=1}^{30} \lg(i+3) \cdot e^{\sqrt{i}}$	26	$\sum_{j=1}^{15} \frac{\sqrt{\lg(j+2)}}{j+2}$
27	$\sum_{k=1}^{20} \sqrt{\operatorname{tg} \frac{1}{k}}$	28	$\sum_{l=1}^{25} \frac{0,5 + \lg l}{\arccos 0,5/l}$
29	$\sum_{m=1}^{30} \cos 10^{-0,1m}$	30	$\sum_{n=1}^{35} \frac{1}{\sqrt{ \operatorname{tg}(n+1) }}$

Задание 2.2. Двойные и кратные циклы

Целью работы является освоение программирования алгоритмов с вложенными циклами. Пример такой задачи – вычисление двойной суммы.

Пример задания.

Вычислить сумму $\sum_{i=1}^{10} \sum_{j=1}^{\infty} \frac{j^{1/i}}{j!}$ с точностью до 10^{-4} .

Здесь внешней суммой является сумма по i , а внутренней – сумма по j . Можно рассматривать вычисление этих сумм отдельно, учитывая что вычисление внутренней суммы является частью вычисления внешней суммы, то есть телом внешнего цикла.

Сами суммы в задании никак не обозначены, и им необходимо присвоить имена, которые могут быть относительно произвольные: **SUM_IN** (внутренняя) и **SUM_OUT** (внешняя); **SUMI** и **SUMO**; **S1** и **S2** и другие. Так же обозначается и внутреннее слагаемое: **SLAG**, **S** и так далее.

Внешняя сумма по i вычисляется так же, как и в задании 2.1. Так как используется счетчик циклов, и количество слагаемых определено, то для ее вычисления справедлив алгоритм, представленный на рис. 12, кроме блока 4. Блок 4 – это не обычное сложение, а внутренний цикл по вычислению суммы по j . Таким образом, блок 4 замещается несколькими блоками, производящими вычисление внутренней суммы. Говорят, что цикл (суммирование) по j вложен в цикл по i .

Внутренняя сумма по j является примером суммы с бесконечным верхним пределом. Проверка окончания цикла осуществляется следующим образом. Так как выражение под знаком суммы постепенно убывает с ростом слагаемых в сумме, то наступает момент, когда очередное слагаемое станет меньше наперед заданного числа ϵ (грубо говоря, точности вычисления сумм), и остальные слагаемые будут мало влиять на конечный результат. Поэтому, когда выражение под знаком суммы $|f(j)|$ будет меньше ϵ , то вычисления прекращаются и предполагается, что сумма найдена с заданной точностью. Если возможны отрицательные значения под знаком суммы, то обязательно надо использовать **функцию вычисления модуля ABS**, так как, например, число -10 меньше 10^{-4} (0.0001), но заданная точность еще не достигнута.

Так как количество слагаемых заранее неизвестно, то циклом **FOR** пользоваться нельзя. Для этих целей предназначаются циклические операторы **WHILE** (соответствует блок-схеме, приведенной

на рис. 11) и REPEAT. Необходимо помнить, что у них параметр цикла автоматически не изменяется и его надо менять принудительно, а при составлении блок-схемы алгоритма вычисления внутренней суммы (вместо блока 4, рис. 12.) блок «Модификация» не используется.

При вычислении суммы в некоторых вариантах должен вычисляться факториал, который определяется формулой

$$j! = \prod_{m=1}^j m .$$

Где Π – знак произведения (аналогично знаку суммы), то есть $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$. Факториал можно вычислить отдельным циклом, третьим, а можно и прямо в цикле вычисления внутренней суммы. Для этого вводится дополнительная переменная, например $f = j!$, и затем в цикле f умножается на текущее значение j .

В языке Турбо Паскаль под переменные типа **INTEGER** выделяется два байта, и допустимые для них значения находятся в диапазоне только от **-32768** до **32767**. Поэтому число $10!$, реально равное **3628800**, в этом случае будет представлено как **24320**. Таким образом выражение под знаком суммы может никогда и не стать меньше заданной точности. Для работы с большими целыми числами (факториалом) во избежание переполнения рекомендуется использовать вещественный тип **Real** с диапазоном представления от $2.9 \cdot 10^{-39}$ до $1.7 \cdot 10^{38}$, или целый тип **LongInt** с диапазоном от **-2.147.483.648** до **2.147.483.647**.

Кроме значения суммы при отладке полезно вывести значение счетчика циклов, то есть узнать, из скольких слагаемых состоит каждая сумма. Например, если все или некоторые внутренние суммы состоят из одного слагаемого, то в программе явно логическая ошибка.

Требование данной работы: при составлении программы точность обязательно должна задаваться в форме вещественного числа с плавающей точкой, а вывод суммы выполняться с фиксированной точкой и количеством знаков дробной части, соответствующим заданной точности.

Варианты заданий контрольной работы приведены в табл. 4.

Таблица 4. Варианты заданий 2.2

№ вар.	Вычислить	Точность вычислений ϵ
1	$3 \sum_{i=1}^{10} \sum_{j=1}^{\infty} \frac{0,3ij}{i^3 + j^3}$	10^{-3}
2	$\sum_{i=1}^{10} i^2 \sum_{j=1}^{\infty} \frac{i+j}{j!}$	10^{-4}
3	$0,3 \sum_{i=1}^{20} \sum_{j=1}^{\infty} \frac{\sin \frac{ij}{p}}{i^5 + j^5}$	10^{-3}
4	$\sum_{i=1}^{15} \cos \frac{p}{i} \sum_{j=1}^{\infty} \frac{\sqrt{i+j}}{(i \cdot j)^5}$	10^{-4}
5	$8 \sum_{i=1}^{15} \sum_{j=1}^{\infty} \frac{\sin \frac{p}{ij}}{j!}$	10^{-3}
6	$\sum_{i=1}^{10} e^{-i} \sum_{j=1}^{\infty} \frac{i^{0,1j}}{j!}$	10^{-4}
7	$\sum_{i=1}^{15} i \sum_{j=1}^{\infty} \frac{\cos ij}{i^3 + j^4}$	10^{-3}
8	$2,7 \sum_{i=1}^{20} \ln i \sum_{j=1}^{\infty} \frac{\sqrt{i+j}}{j!}$	10^{-4}
9	$\sum_{i=1}^{12} i^2 \sum_{j=1}^{\infty} \frac{\operatorname{tg} \frac{1}{ij}}{j^3}$	10^{-3}
10	$\sum_{i=1}^{10} \cos 2i \sum_{j=1}^{\infty} \frac{i^2 + 0,5j}{j!}$	10^{-4}

1	2	3
11	$1,4 \sum_{i=1}^8 10^{-i} \sum_{j=1}^{\infty} \frac{e^{-i/j}}{i^3 + j^2}$	10^{-3}
12	$\sum_{i=1}^6 \sum_{j=1}^{\infty} \frac{j + \sin 0,3ij}{i - j^4 + 2}$	10^{-4}
13	$\sum_{i=1}^8 i \sum_{j=1}^{\infty} \frac{i - 2j - 4,3}{(ij)^3}$	10^{-3}
14	$6,4 \sum_{i=1}^{10} \frac{1}{i+1} \sum_{j=1}^{\infty} \frac{e^{i/j}}{i^2 + j^2}$	10^{-4}
15	$\sum_{i=1}^5 e^{-i} \sum_{j=1}^{\infty} \frac{\sqrt{i^2 + j^2 + 1}}{j!}$	10^{-3}
16	$\sum_{i=1}^7 \sum_{j=1}^{\infty} \frac{\sqrt{i+2}\sqrt{j}}{ij^3}$	10^{-4}
17	$2,3 \sum_{i=1}^9 \sqrt{i} \sum_{j=1}^{\infty} \frac{\arcsin \frac{1}{ij+1}}{j^4 + i^2 + 1}$	10^{-3}
18	$\sum_{i=1}^6 i^{0,4} \sum_{j=1}^{\infty} \frac{10^{0,1ij}}{j! - i^2 + 5,2}$	10^{-4}
19	$\sum_{i=1}^8 i^2 \sum_{j=1}^{\infty} \frac{i + \ln ij}{\sqrt{i+j}^3}$	10^{-3}
20	$7 \sum_{i=1}^{10} \sum_{j=1}^{\infty} \frac{\operatorname{tg} \frac{1}{i} + \operatorname{tg} \frac{1}{j}}{j!}$	10^{-4}

1	2	3
21	$\sum_{i=1}^5 \frac{1}{\sin i} \sum_{j=1}^{\infty} \frac{\sqrt{i^2 + j^2}}{(i+j)^{3,8}}$	10^{-3}
22	$\sum_{i=1}^7 \operatorname{tg} i \sum_{j=1}^{\infty} \frac{1}{j^3 - i^2 + 10,1}$	10^{-4}
23	$0,15 \sum_{i=1}^9 (i-0,3) \sum_{j=1}^{\infty} \frac{\sqrt{i \cdot j}}{i^3 + 4j^2 + 4i \cdot j}$	10^{-3}
24	$\sum_{i=1}^6 \sum_{j=1}^{\infty} \frac{i^{0,1j}}{j!}$	10^{-4}
25	$\sum_{i=1}^8 0,1i \sum_{j=1}^{\infty} \frac{i + j^2 - 3,2}{3j^4 + 2ij + 1}$	10^{-3}
26	$3,3 \sum_{i=1}^{10} \frac{1}{i+p} \sum_{j=1}^{\infty} \frac{0,8p^2 j - i}{i^5 + j^5}$	10^{-4}
27	$\sum_{i=1}^5 \sqrt{i} \sum_{j=1}^{\infty} \frac{\ln ij}{j! + i}$	10^{-3}
28	$\sum_{i=1}^7 \sum_{j=1}^{\infty} \frac{\sqrt{i+j}}{i^5 + j^4 + 2ij}$	10^{-4}
29	$5 \sum_{i=1}^9 \cos i \sum_{j=1}^{\infty} \frac{i}{j^3 + 1}$	10^{-3}
30	$\sum_{i=1}^6 i \sum_{j=1}^{\infty} \frac{0,1(e^{0,55} i + j)}{j!}$	10^{-4}

ПРИЛОЖЕНИЕ А

Таблица 5. Основные команды интегрированной
инструментальной оболочки
Турбо Паскаль 7.0

Команда меню	«Горячая клавиша»	Функция
-	F10	Активизация строки меню
-	<Alt+F10>	Вызов локального меню
-	<Alt+Литера>	Открытие озаглавленного выбранной литерой подчиненного меню из строки меню
New	-	Создание нового окна с именем NONAMEnn.PAS для ввода программы
File/ Open	F3	Открытие нового окна и загрузка в него выбранного файла
File/ Save	F2	Сохранение на диске файла из активного окна. Если было создано новое окно, выполнится команда Save as
File/ Save as	-	Сохранение на диске файла из активного окна с запросом на новое имя файла
File/ Cnange dir	-	Смена текущей директории
File/ Exit	<Alt+X>	Завершение сеанса работы с Турбо Паскалем с сохранением (после подтверждением) файлов, измененных редактором текста
Edit/ Undo	<Alt+Backspace>	Отмена всех изменений в текущей строке
Edit/ Cut	<Shift+Del>	Перенос выделенного блока в буфер промежуточного хранения
Edit/ Copy	<Ctrl+Ins>	Копирование блока в буфер промежуточного хранения

Продолжение табл. 5

1	2	3
Edit/ Paste	<Shift+Ins>	Копирование блока из буфера промежуточного хранения в окно редактирования
Edit/ Clear	<Ctrl+Del>	Удаление выделенного блока
Run/ Run	<Ctrl+F9>	Компиляция и выполнение программы под управлением интегрированной инструментальной оболочки
Run/ Step over	F8	Трассировка программы пооператорно с выполнением подпрограмм без пооператорной детализации
Run/ Trace Into	F7	Трассировка программы пооператорно с пооператорным выполнением всех подпрограмм
Run/ Go to Cursor	F4	Выполнение программы, расположенной в активном окне, до позиции курсора
Run/ Program Reset	<Ctrl+F2>	Установка программного счетчика на начало программы и закрытие всех ранее открытых программой файлов
Compile/ Compile	<Alt+F9>	Компиляция программы из активного окна
Compile/ Make	F9	Компиляция и редактирование связей программы
Compile/ Destination Memory/Disk	-	Место компиляции программы: либо в оперативную память, либо на диск
Debug/ Call Stack	<Ctrl+F3>	Открытие окна протокола используемых процедур
Debug/ User Screen	<Alt+F5>	Переключение на пользовательский экран
Debug/ Evaluate/ Modify	<Ctrl+F4>	Просмотр и изменение значений переменных

1	2	3
Debug/ Add Watch	<Ctrl+F7>	Дополнение списка переменных, наблюдаемых в Watch -окне
Option/ Save TURBO.TP	-	Сохранение конфигурации ИИО в файле TURBO.TP
Window/ Size/ Move	<Ctrl+F5>	Изменение положения и размера окна
Window/ Zoom	F5	Изменение (увеличение/ уменьшение) размера активного окна
Window/ Next	F6	Переход к следующему окну
Window/ Previous	<Shift+F6>	Возврат к предыдущему окну
Window/ Close	<Alt+F3>	Закрытие активного окна
Window/ List	<Alt+O>	Вызов окна, в котором содержится список всех открытых окон
-	<Alt+ Цифра >	Переход к окну с указанным номером
-	F1	Активизация окна контекстно-зависимой помощи
Help/ Previous topic	<Alt+F1>	Возврат к предыдущей справке
Help/ Topic search	<Ctrl+F1>	Активизация синтаксической справки, то есть справки об операторе, на который указывает маркер
Help/ Index	<Shift+F1>	Вызов содержания справочной подсистемы

ПРИЛОЖЕНИЕ Б

Таблица 6. Команды редактора текста. Управление курсором

Клавиши	Действие
Стрелка вверх	Курсор переводится на одну строку вверх
Стрелка вниз	Курсор переводится на одну строку вниз
Стрелка влево	Курсор переводится на одну позицию влево
Стрелка вправо	Курсор переводится на одну позицию вправо
<Ctrl+Стрелка влево>	Курсор переводится на одно слово влево
<Ctrl+Стрелка вправо>	Курсор переводится на одно слово вправо
Home	Курсор переводится на начало строки
End	Курсор переводится в конец строки
<Ctrl+Home>	Курсор переводится на первую строку экрана
<Ctrl+End>	Курсор переводится на последнюю строку экрана
PgUp	Продвижение по файлу на одну страницу назад
PgDn	Продвижение по файлу на одну страницу вперед
<Ctrl+PgUp>	Курсор переводится в начало файла
<Ctrl+PgDn>	Курсор переводится в конец файла
<Ctrl+W>	Экран сдвигается «вверх» по тексту (при этом курсор неподвижен)
<Ctrl+Z>	Экран сдвигается «вниз» по тексту (курсор неподвижен)
<Ctrl+Q>+B	Курсор переводится в начало блока
<Ctrl+Q>+K	Курсор переводится в конец блока
<Ctrl+Q>+P	Курсор перемещается на исходную позицию после поиска

1	2
<Ctrl+P>	Ввод специального символа
Вставка и удаление	
Del	Удаление символа, указываемого курсором
Ins	Переключение между режимами вставки и замены
BackSpace	Удаление символа слева от курсора
<Ctrl+T>	Удаление слова справа от курсора
<Ctrl+Q>+Y	Удаление части строки от курсора до конца строки
<Ctrl+Y>	Удаление строки, указываемой курсором
<Ctrl+Q>+L	Восстановление строки, удаленной комбинацией клавиш <Ctrl+Y>, в том месте текста, где она была расположена
<Ctrl+N>	Вставка строки
Операции с блоками	
<Shift+стрелки>	Расширение маркируемого блока
<Ctrl+K>+B	Указание начала маркируемого блока
<Ctrl+K>+K	Указание конца маркируемого блока
<Ctrl+K>+T	Маркирование слова
<Ctrl+K>+H	Снятие/ восстановление маркировки
<Ctrl+K>+I	Сдвиг маркированного блока вправо
<Ctrl+K>+U	Сдвиг маркированного блока влево
<Ctrl+K>+C	Копирование маркированного блока в то место, где установлен курсор
<Ctrl+K>+V	Перенос маркированного блока в то место, где установлен курсор

1	2
<Ctrl+Ins>	Копирование маркированного блока в буфер промежуточного хранения (Edit/ Copy)
<Shift+Del>	Перенос маркированного блока в буфер промежуточного хранения (Edit/ Cut)
<Shift+Ins>	Копирование маркированного блока из буфера промежуточного хранения в то место, где установлен курсор (Edit/ Paste)
<Ctrl+K>+Y	Удаление маркированного блока
<Ctrl+K>+P	Печать маркированного блока (File/ Print)
<Ctrl+K>+R	Вставка текста из файла в позицию, указываемую курсором (Read)
<Ctrl+K>+W	Запись блока в файл (Write)
<Ctrl+K>+F	Переключатель режима заполнения, позволяющий оптимизировать заполнение интервалов между словами (пробелами/ табуляциями) (Options/ Env./ Editor/ Optimal Fill)

1	2
Поиск и замена	
<Ctrl+Q>+F	Поиск указанной строки (Search/ Find). Для указания опций открывается специальное окно
<Ctrl+Q>+A	Поиск указанной строки и замена(Search/ Replace). Для указания опций открывается специальное окно
<Ctrl+Q>+[Поиск разделителя ({, [, (, ",'), парного по отношению к указываемому курсором (удобно искать границы комментариев)
<Ctrl+Q>+]	Поиск разделителя ([, },), ",'), парного по отношению к указываемому курсором
<Ctrl+K>+<n>	Установка отметки в тексте. n - цифра от 0 до 9
<Ctrl+Q>+<n>	Перевод курсора на отметку в тексте. n - цифра от 0 до 9
<Ctrl+Q>+W	Перевод курсора на позицию, где дано сообщение об ошибке
<Ctrl+L>	Продолжение поиска/ замены с установленными ранее опциями
<Ctrl+U>	Прерывание поиска/ замены

Список литературы

1. Зуев Е.А. Язык программирования Turbo Pascal 6.0, 7.0. – М.: Веста, Радио и связь, 1993.
2. Климова Л.М. PASCAL 7.0: Практическое программирование. Решение типовых задач.: Учебное пособие. – М.,: 2000.
3. Немюгин С.А. TURBO PASCAL – СПб, 2000.
4. Турбо Паскаль 7.0. Самоучитель. – СПб.: Питер; К.: Издательская группа BHV, 2002. – 416 с.: ил.
5. Фаронов В.В. Турбо Паскаль 7.0. Начальный курс. Учебное пособие. – М.: «Нолидж», 1999. – 616 с., ил.
6. Федоренко Ю. Алгоритмы и программы на Turbo Pascal. Учеб. Курс. – СПб.: Питер, 2001.
7. Единая система программной документации. ГОСТ 19.701-90 (ИСО 5807-85). Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

