МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский национальный технический университет

Кафедра «Горные машины»

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

Лабораторный практикум

Часть 2

Минск
БНТУ
2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Горные машины»

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

Лабораторный практикум

В 3 частях

Часть 2

ПРОГРАММИРОВАНИЕ ТИПОВЫХ АЛГОРИТМОВ

Рекомендовано учебно-методическим объединением по образованию в области горнодобывающей промышленности

Минск
БНТУ
2018

УДК 004.438(076.5)
ББК 22.18я7
П78

Составитель
С. М. Петренко

Рецензенты:
Ю. А. Шпургалов, Ю. И. Тарасов

П78 **Программирование** в среде Turbo Pascal 7.0: лабораторный практикум: в 3 ч. / сост. С. М. Петренко. – Минск: БНТУ, 2014– .– Ч. 2: Программирование типовых алгоритмов. – 2018. – 46 с.
ISBN 978-985-550-538-0 (Ч. 2).

Практикум содержит лабораторные работы для практического освоения начальных навыков программирования на алгоритмическом языке Турбо Паскаль при изучении дисциплины «Информатика» студентами специальностей горного профиля.

Часть 1 «Интегрированная среда программирования TURBO 7.0» вышла в 2014 г. (сост.: С. М. Петренко, И. М Ковалёва).

УДК 004.438(076.5)
ББК 22.18я7

ISBN 978-985-550-538-0 (Ч. 2)
ISBN 978-985-550-354-6

ВВЕДЕНИЕ

Лабораторный практикум предназначен для пользователей (в первую очередь для студентов-заочников), которые только начинают осваивать алгоритмический язык и систему программирования Турбо Паскаль 7.0. Поэтому задания на каждую лабораторную работу имеют вводную информацию, которая содержит минимально необходимые сведения и примеры программного кода по рассматриваемой теме. Зарезервированные слова алгоритмического языка Турбо Паскаль в примерах и пояснениях выделены жирным шрифтом.

Основное назначение практикума – получение пользователями практических навыков разработки типовых алгоритмов и составления по ним программ на алгоритмическом языке Турбо Паскаль.

В практикум включены лабораторные работы по обязательным в любой программе действиям – объявлению используемых в программе констант и переменных, организации ввода и вывода, которые являются базовыми, но почему-то не рассматриваются в известных лабораторных практикумах.

Программа решения реальной прикладной задачи является комбинацией ряда типовых вычислительных процессов: линейных, разветвляющихся или циклических. Данный лабораторный практикум содержит соответствующие лабораторные работы по реализации этих типовых алгоритмов средствами алгоритмического языка Турбо Паскаль.

В заданиях на лабораторные работы используются простые типы данных. Программирование с использованием структурированных типов данных будет рассмотрено в следующей части практикума.

В инструкциях каждой лабораторной работы указано содержание отчета. Отчеты оформляются на листах бумаги формата А4 в соответствии с требованиями к отчетам по лабораторным работам стандарта БНТУ (Единая система учебной документации. Отчет о лабораторной работе. Общие требования и правила оформления: СТП 10-02.01-87. – Минск, 1987).

Лабораторная работа № 1

ТИПЫ ДАННЫХ

Цель работы: практически освоить порядок описания используемых в Паскаль-программе констант и переменных; исследовать совместимость простых типов данных.

Вводная информация

Разделы объявлений в структуре Паскаль-программы

Программный блок исходной Паскаль-программы в общем случае может состоять из пяти *разделов описаний* (описаний меток, констант, переменных, процедур и функций, определения созданных пользователем типов данных) и *раздела операторов* – основного блока программы. Разделы описаний записываются *до раздела операторов.*

Константы и переменные, используемые в исходной Паскаль-программе, должны быть предварительно описаны (объявлены) в соответствующих разделах описаний программного блока.

Разделы описаний начинаются соответствующими зарезервированными словами алгоритмического языка, затем следуют имена используемых в программе элементов и их значения либо типы.

Если при компиляции исходной программы будет обнаружена необъявленная константа или переменная, процесс компиляции будет прерван с выводом сообщения об ошибке **3 Unknown identifier** (неизвестный идентификатор).

Типы данных

Для переменных Паскаль-программы указание типа обязательно. Под типом понимается идентификатор типа (в дальнейшем – имя_типа), в качестве которого используется один из типов алгоритмического языка Турбо Паскаль, определенных разработчиком, или тип, который задается пользователем в разделе описаний типов **Type**.

Задание типа данных определяет:

– диапазон возможных значений констант, переменных, функций и результатов вычисления выражений, принадлежащих к этому типу;

– форму представления данных этого типа в ЭВМ и размер занимаемой памяти;

– операции, которые могут выполняться над данными этого типа.

Стандартные типы данных включают:

– целочисленные типы;

– вещественные типы;

– логический тип;

– символьный тип.

Для описания положительных целых величин используются типы **byte** и **word**, для целых со знаком – **shortInt, integer** и **longInt**. Для описания вещественных величин используются типы **single, real, double** и **extended**.

Тип **comp** может содержать только целые числа от –263 (+1) до +263 (–1), но эти числа хранятся в памяти в вещественном формате, поэтому тип **comp** считается вещественным. Данные типа **comp** обрабатываются так же, как и данные других вещественных типов, но дробная часть числа при этом автоматически отбрасывается.

Логический тип обозначается зарезервированным словом **boolean**. Данные этого типа могут принимать только значения true (истинно) или false (ложно).

Символьный тип обозначается зарезервированным словом **char**. Данные этого типа могут принимать значения только одного символа (знака, буквы или кода).

Для создания собственного типа пользователя используется конструкция:

Type
имя_типа1 = определение_типа1;
имя_типа 2 = определение_типа 2; и т. д.

В качестве «определения_типа» здесь используются стандартные типы данных, определенные разработчиком алгоритмического языка Турбо Паскаль, или типы пользователя.

Типы данных, определенные разработчиком алгоритмического языка Турбо Паскаль, делятся на *простые* и с*ложные*.

Простые типы данных, в свою очередь, делятся на *ограниченные* и *скалярные*. Скалярные типы включают в себя *перечислимые* и *стандартные*.

Ограниченный тип применяется для данных, которые принимают значения из некоторого ограниченного диапазона. При определении ограниченного типа указываются граничные константы, определяющие начальное и конечное значения диапазона. Граничные константы разделяются двумя точками. Описание ограниченного типа имеет следующий вид:

Type
имя_типа = левая константа..правая константа;

В качестве граничных констант можно использовать значения любого простого типа, кроме вещественного. Обе граничные константы должны быть одинакового типа, левая граничная константа должна быть меньше правой.

Например:

Type
Diapozon = –10..10;
Letter = 'A'..'Z';

Перечислимый тип используется для данных, которые могут принимать любое значение из строго ограниченного количества значений из списка. Под списком понимается перечень констант, разделенных запятыми.

Описание перечислимого типа имеет следующий вид:

Type
имя_типа = (список констант);

Например,

Type Ozenka = (Seven, Six, Five, Four);

Список констант-значений записывается в круглых скобках в строго определенном порядке. Каждая из констант имеет поряд-

ковый номер, который начинается с нуля. Упорядоченность констант позволяет сравнивать их с помощью операций отношений (<, <=, =, <>, >=, >). При сравнении переменной со значением Seven и переменной со значением Four последняя будет больше, так как больше ее порядковый номер.

К сложным типам данных относятся строки символов, массивы, записи, множества, файлы (три вида).

Простейший из сложных типов – строковый, который используется для данных в виде последовательности символов, для него зарезервированные слова **string** (строка максимальной длины) или **string[*n*]**, где *n* – количество символов в строке.

Другие сложные типы строятся на базе простых и будут рассмотрены в следующей части лабораторного практикума.

Описание констант

В начале раздела описаний констант записывается зарезервированное слово **Const**, далее следуют идентификатор константы (в дальнейшем – имя_константы), знак «=» и значение константы:

```
Const
имя_константы1 = значение 1;
имя_константы2 = значение 2; и т. д.
```

Пример описания констант:

```
Const
a = 13;              {целая числовая константа}
d = 0.12345E+3;      {вещественная числовая константа}
b = TRUE;            {логическая константа}
ch = ‘Ф’;            {символьная константа}
s = ‘студент’;       {строковая константа}
```

В процессе выполнения программы заданные значения объявленных таким образом констант не изменяются.

Если начальные значения, заданные при описании констант, необходимо изменять в процессе выполнения программы, используются типизированные константы, которые можно рассматривать

как переменные с заданными в начале выполнения программы значениями.

Описание типизированных констант:

Const
имя_константы1: Тип1 = Значение 1;
имя_константы2: Тип2 = Значение 2; и т. д.

Пример описания типизированных констант:

Const
Imax: **integer** = 1234;
Omega: **real** = 3.45;

Использование типизированных констант при описании других типов, задаваемых пользователем, запрещено.

Описание переменных

Переменная – это именованный элемент программы, который имеет идентификатор (далее – имя) и значение определенного типа. Имя служит для обращения к области памяти, в которой хранится значение переменной. В процессе выполнения программы значение переменной может изменяться.

Форма описания переменных:

Var
список1: имя_типа1;
список2: имя_типа 2; и т. д.

Список может содержать одно имя переменной или разделенные символами «запятая» имена нескольких переменных одинакового типа. В качестве имени типа используются зарезервированные слова-идентификаторы стандартных скалярных типов или имена типов, объявленных в разделе *Type*.

Примеры описания переменных:

```
Type
  Diapozon = –10..10;
  Ozenka = (Seven, Six, Five, Four);
Var
  a, p, t: byte;
  x, y: real;
  b1: Boolean;
  c1: char;
  element1: diapozon;
  result:ozenka;
  s1, s2: string[10];
```

Операция присваивания и совместимость типов и значений

Чтобы поместить значение в объявленную в программе переменную, используется операция присваивания:

Имя_переменной:= Значение;

Составной символ «:=» называется оператором присваивания. Смысл операции присваивания – Имя_переменной начинает хранить Значение, стоящее справа. В качестве Значения может выступать конкретное значение переменной, результат вычисления выражения или имя другой переменной:

```
P:= 3;
X:= 2.35;
A:= A + 1;
Element:= P;
```

Для выполнения операции присваивания необходимо, чтобы типы Имя_переменной и Значение были совместимыми. При несовместимости типов компиляция программы прерывается и выводится сообщение об ошибке **26 Type mismatch** (несоответствие типов).

Задание на лабораторную работу

Объявить в программе константы и переменные для одного из вариантов значений из табл. 1.1 и 1.2.

Таблица 1.1

Значения констант и переменных интервального и перечислимого типов

№ вар.	Константа	Типизированная константа	Диапазон значений переменной	Возможные значения переменной
1	Студент	–1,1	–15...55	Idn1, Idn2, Idn3
2	Группа	2,2	12...53	Elem1, elem2, elem3
3	ФГДЭ	–3,33	–99...11	Zima, Vesna, Leto, Osen
4	Сессия	4,51	‘a’...’t’	Kot, dog, mouse
5	Зачет	–5,18	‘б’...’ф’	Sm1, sm2, sm3
6	Экзамен	6,37	66...87	Elm1, elm2, elm3
7	Alpha	–7,77	‘г’...’ю’	Zima, vesna, leto
8	Beta	8,39	13...27	Slon, lev, kot
9	Gamma	121	47...89	Al1, bet2, gam3
10	13,5	–13	‘д’…’я’	Razn1, Razn 2, Razn3, Razn4
11	–123	0,0032	–50…50	Vesna, Leto, Zima, Osen
12	5,67	–16,17	98…106	Obj1, Obj2, Obj3, Obj4
13	1000	18,96	‘p’...’z’	Kol, Kol1, Kol 2
14	–0,00456	19,9	–3...44	Str0, Str1, Str2, Str3, Str4
15	12,8	–65	–27...37	Per1, Per2, Per3

Таблица 1.2

Значения переменных

№ вар.	I	j	k	x	y	b1	ch	s
1	–123	66	71426	0,0032	–6,28	TRUE	Й	Alpha
2	–133	19	53678	–16,17	–999,77	FALSE	Ц	Beta
3	30246	–3456	69147	18,96	1234,56	TRUE	Н	Gamma
4	254	555	–6895	19,9	17,865	FALSE	Г	ГОРНЫЕ
5	107	–127	127	–7,77	56,432	TRUE	Ш	Машины
6	53678	71426	–127	8,39	0,0032	FALSE	Щ	Торф
7	69147	254	555	0,326	–16,17	TRUE	З	Порода
8	–6895	30246	–3456	–11,22	18,96	FALSE	Д	Минерал
9	71426	254	19	638,927	19,9	TRUE	Ж	Допуск
10	127	107	254	3,14	–7,77	FALSE	Э	Студент
11	–127	53678	107	–6,28	8,39	TRUE	Л	Группа
12	555	–133	53678	–999,77	0,326	FALSE	Ы	ФГДЭ
13	–3456	254	–123	1234,56	–11,22	TRUE	Ф	Сессия
14	19	–6895	–133	17,865	638,927	FALSE	Я	Зачет
15	66	–123	30246	56,432	3,14	TRUE	Ч	Экзамен

Присвоить им значения по варианту задания.

Используя взаимное переприсваивание, исследовать совместимость переменных разных типов. Если типы не совместимы, заключить эту строку программы в фигурные скобки (закомментировать), добавив напротив оператора присваивания примечание – несовместимы:

{ p:= x; несовместимы}.

Содержание отчета

1. Цель работы.
2. Вариант задания на лабораторную работу.
3. Распечатка текста программы.
4. Выводы (о совместимости типов).

Лабораторная работа № 2

ВВОД-ВЫВОД ДАННЫХ

Цель работы: практически освоить процедуры ввода-вывода данных.

Вводная информация

Обмен данными с внешними файлами и периферийными устройствами

В обмене данными в Паскале участвуют источник, канал передачи и приемник. При *вводе* данных источником является файл, а приемником – ОЗУ. При *выводе*, наоборот, данные из ОЗУ пересылаются в файл. Под файлом понимается внешний (создаваемый вне программы) файл данных на каком-либо носителе или аппаратное устройство, имеющее в Турбо-системе статус файла (клавиатура, дисплей, принтер, параллельные и последовательные порты, фиктивное устройство).

Одним из типов внешних файлов является текстовый файл, в котором информация хранится в виде последовательностей обычных символов подобно тому, как они записываются на бумаге. Символы в текстовом файле разделяются на строки различной длины, которые заканчиваются специальным символом признака конца строки.

Содержимое таких файлов можно просматривать в окне редактора Турбо-системы Edit как обычный текст, поэтому текстовые файлы удобно использовать для вывода в них результатов работы Паскаль-программы.

Формат описания текстового файла в разделе описаний программы:

Var имя_файла: **text;**

Пример описания текстового файла:

```
Var
fp: text; {здесь fp – файловая переменная (логическое имя файла)}
```

В программном блоке необходимо выполнить процедуру связывания логического имени файла с физическим файлом на диске:

```
BEGIN
Assign (fp, 'E:\ 102812\ Rez1. pas'); {связывание файла с логическим именем fp с физическим файлом на диске с полным именем 'E:\102812\ Rez.pas}
……
……
END.
```

После выполнения процедуры **Assign** все операции, предписанные в Паскаль-программе файлу с логическим именем fp, будут выполняться над физическим файлом Rez.pas.

Типовые операции над текстовым файлом с логическим именем fp:

Reset (fp) – открытие текстового файла с логическим именем fp (физического файла Rez1.pas) для чтения из него данных, начиная с начала первой строки.

Read (fp, y1, y2, …, y*n*) – чтение данных из открытого файла fp, прочитанные значения последовательно присваиваются переменным y1, y2, …, y*n* списка ввода без перехода в начало следующей строки.

Readln (fp, список_ввода) – то же, что и в предыдущем случае, но после чтение данных выполняется переход на новую строку (следующая процедура ввода будет вводить данные из новой строки).

Readln (fp); – переход в начало новой строки в файле без чтения данных.

Rewrite (fp); – открытие файла fp для перезаписи в него с начала первой стоки, предыдущие данные стираются.

Append (fp); – открытие ранее созданного текстового файла fp для дозаписи данных в конец файла, предыдущие данные сохраняются.

Write (fp, x1, x2, … x*n*); – запись в текстовый файл fp значений переменных x1, x2, … x*n* списка вывода без перехода в начало следующей строки.

Writeln (fp, список_вывода); – то же, что и в предыдущем случае, но с завершением записи в текущую строку и переходом в начало новой строки.

Writeln (fp); – переход в начало новой строки без записи данных (создание «пустой» строки в текстовом файле.

Close (fp); – закрытие файла).

Ввод данных с клавиатуры

Если в процедурах **Read** или **Readln** перед списком ввода не указывается логическое имя файла, то по умолчанию, принятому в Турбо-системе, файлом-источником является клавиатура. Возможны три варианта записи:

Read (список_переменных); – каждое вводимое с клавиатуры значение последовательно присваивается переменным, имена которых перечислены в списке ввода.

Readln (список_переменных); – то же, но после ввода выполняется переход на новую строку на дисплее.

Readln; – переход на новую строку без ввода данных.

Примечание. При выполнении процедур **Read** или **Readln** Турбо-система переходит в режим ожидания ввода данных (окно с черным цветом фона) и остается в нем до момента ввода последнего значения. Поэтому перед использованием этих процедур следует выводить на дисплей текстовую подсказку пользователю – в какой последовательности вводить значения переменных:

Write (‘Введите x, y, z’); {Вывод на дисплей строки «Введите x, y, z»}

Readln (x, y, z); {Присваивание вводимых значений переменным x, y, z}

В качестве разделителя вводимых значений используются:

– при вводе значений в одной строке – один или несколько пробелов;

– при вводе значений по-отдельности – признак конца строки (нажатие клавиши Enter).

Вывод данных на дисплей и в текстовый файл

Для вывода данных на дисплей используются процедуры **Write** или **Writeln** без указания логического имени файла перед списком вывода. Элементами списка вывода могут быть либо имена констант, переменных или результат математического выражения, либо

символьные последовательности, заключенные в апострофы, либо их комбинация. В качестве разделителя элементов списка вывода используется символ «запятая»:

Writeln (‘Исходные данные:’); {Элемент списка – текст в апострофах}
Writeln (‘x =’, x, ‘y =’, y, ‘z =’, z); {Элементы списка – текст и имена переменных}

Последовательности символов, заключенные в апострофы, выводятся без изменений. Вместо имен переменных выводятся соответствующие текущие значения этих переменных.

При использовании в списке вывода имен переменных без указания формата значения переменных выводятся с плавающей десятичной точкой. Количество позиций для вывода значений переменных устанавливается Турбо-системой по умолчанию.

Для форматного вывода значений переменных используются форматы:

– P:M, где P – имя переменной; M – целое число, задающее количество позиций для вывода значения P;

– P:M:N, где P – имя переменной; M – общее число позиций для вывода значения P, включая знак числа, десятичную точку и дробную часть;

– N – целое число, задающее количество позиций для вывода дробной части.

Пример форматного вывода на дисплей:

Writeln (‘ x =’, x:5:2, ‘ y =’, y:7:3, ‘ z =’, z:7);

Обычно формат P:M используется для вывода целых чисел, при этом избыточные позиции заменяются символами пробела перед числом. При использовании этого формата для вывода вещественных чисел по умолчанию реализуется представление числа с плавающей десятичной точкой.

При выводе вещественных чисел по формату P:M:N избыточные позиции в целой части заменяются символами пробела перед числом, а избыточные позиции в дробной части заполняются нулями в конце дробной части.

При выполнении процедуры **Writeln**; выполняется переход в начало новой строки (разделение выводимых на дисплей строк «пустой» строкой).

Пример вывода данных в текстовый файл (перед списком вывода следует указать логическое имя файла):

Writeln (fp, ‘x =’, x:5:2, ‘y =’, y:7:3, ‘z =’, z:7);.

Процедура **Writeln**(fp); пропускает строку в текстовом файле. Для пропуска нескольких строк следует повторить эту процедуру несколько раз.

Задание на лабораторную работу

Составить программу, которая обеспечивает:

– ввод с клавиатуры значений переменных для указанного преподавателем варианта из табл. 1.1 и 1.2 задания на лабораторную работу № 1;

– вывод значений этих переменных на дисплей и в текстовый файл с указанием имен переменных.

Определить, данные каких типов нельзя ввести с клавиатуры.

Определить, данные каких типов нельзя вывести на дисплей.

Содержание отчета

1. Цель работы
2. Задание на лабораторную работу
3. Распечатка текста программы
4. Выводы

В выводах необходимо указать типы данных, которые нельзя вводить с клавиатуры и выводить на дисплей.

Лабораторная работа № 3

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Цель работы: получить навыки программирования линейных алгоритмов с использованием математических выражений.

Вводная информация

Понятие линейного алгоритма

Алгоритм – это понятное и точное предписание исполнителю (человеку или вычислительной машине) последовательности действий (команд), направленных на решение поставленной задачи.

Алгоритм называется *линейным*, если его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий.

Математические выражения

В языке Турбо Паскаль могут быть реализованы два вида математических выражений – арифметические или логические. Результат арифметического выражения – число, результат логического выражения – TRUE (истинно) или FALSE (ложно).

В алгоритмической записи математические выражения состоят из операндов (констант, переменных и обращений к функциям) и знаков арифметических или логических операций. Большинство операций в языке Турбо Паскаль являются бинарными и содержат как минимум два операнда:

x +y/z {x, y и z – операнды, + и / – знаки арифметических операций}

a **or** b. {a и b – операнды, **or** – знак логической операции «Или»}

Некоторые операции являются унитарными и содержат только один операнд, знак операции в них всегда предшествует операнду:

–y {присваивание отрицательного значения переменной y}

not (a) {логическая операция **not** над значением переменной a}

При вычислениях сначала выполняются операции наивысшего приоритета, затем – более низкого.

Приоритет операций в математических выражениях:

1) **not** (и другие унитарные операции);
2) вычисление значений стандартных функций;
3) *, /, **div**, **mod**, **and**;
4) +, –, **or**, **xor**;
5) операции отношения (<, >, =, <>, <=, >=).

Операции равного приоритета выполняются последовательно слева направо. Для изменения порядка вычислений используются круглые скобки.

Пример вычисления арифметического выражения. Составить программу для вычисления выражения

$$y = \sqrt[3]{\frac{e^{x+a}}{1-\frac{1}{1+x}}} + \mathrm{tg}(a+x)$$

при a = 3,45 и x = 0,514.

```
Program pr1;
Const a = 3.45;
Var x: real;
BEGIN
Write (‘Введите значение x > 0‘); Readln (x);
Y:= exp(1/3 * ln(exp(x + a)/(1 – 1/(1 + x)))) + sin(a + x)/cos(a + x);
Writeln (‘При a =’, a, ‘x =’, x:5:3, ‘ y =, y:7:3)
END.
```

Пример вычисления логического выражения. Составить программу для определения результата логического выражения:

b = (a > 3) and (c <= 5) or not (a – c > x) при a = 1,2; x = 3,3; c = 0,5.

```
Program pr2;
Var
a, x, c: real; b: boolean;
```

```
BEGIN
a:= 1.2; x:= 3.3; c:= 0.5;
b:= (a > 3) and (c <= 5) or not (a – c > x);
Writeln (‘При a =’, a, ‘c =, c ‘,x =’, x, ‘результат b =’, b)
END.
```

Пояснения. При заданных значениях переменных логическое выражение

(1.2 > 3.3) **and** (0.5 <= 5) **or not** (1.2 – 0.5 < 0).

Результаты операций отношения в скобках:

(FALSE) **and** (TRUE) **or not** (FALSE).

В соответствии с приоритетами последовательность выполнения логических операций будет **not** – **and** – **or**. Тогда: **not** (FALSE) = = TRUE, FALSE **and** TRUE = FALSE, FALSE **or** TRUE = TRUE и результат вычисления логического выражения будет равен TRUE.

Задания на лабораторную работу

Ознакомиться с вводной информацией и выполнить вариант задания, указанный преподавателем.

Задание А.

Разработать алгоритм и составить программу вычисления арифметического выражения по варианту задания из табл. 3.1. Исходные данные вводятся с клавиатуры по запросу программы. На дисплей и в текстовый файл с указанием имен переменных выводятся значения исходных данных и результат вычислений.

Таблица 3.1

Варианты задания А

№ варианта	Выражения	Исходные данные
1	$b = x\sqrt{\left(x^2+a^2\right)^3} + \log_5(x^2+\sqrt{x^2+1})$	a = 2.2, $x > 0$
2	$z = \sqrt{\frac{\left\|x^2-0,1\right\|}{x^2+0,1}} + \frac{x+y^3}{\log_8(1+x^2)}$	$x > 0$, $y > 0$
3	$p = \frac{\lg x - e^{x+y} + x^y}{\sqrt{2}+y^2+\left\|x^3-\ln y\right\|}$	$x > 0$, $y > 0$
4	$p = \frac{\lg x - e^{x+y} + x^y}{\sqrt{2}+y^2+\left\|x^3-\ln y\right\|}$	$x > 0$, $y > 0$
5	$b = 0,1x^5 + e^{-0,1x} - \sqrt{\left\|x\right\|\left\|x+y\right\|}$	$x > 0$, $y > 0$
6	$c = \operatorname{arctg} x - \frac{3}{5}e^{xy} + \frac{(x+y)^x}{(\sin x+1/5)^2}$	$x > 0$, $y > 0$
7	$h = 2 + \frac{x^2}{\sqrt{2}} + \frac{\left\|y^3\right\|}{\sqrt{2}} + \frac{z^4\left(\ln(x)+1\right)\cdot\sqrt{2}}{\sqrt{3}}$	$x > 0$, $y > 0$
8	$f = \frac{\left(\cos x-\sin x\right)^3}{\sqrt{x^2+1}} + \lg^2 xy$	$x > 0$, $y > 0$
9	$f = y^x + \sqrt{\left\|x\right\|+e^y} - \lg\left\|x^2+3x-5\right\|$	$x > 0$, $y > 0$
10	$p = \frac{\lg(x) - e^{x+y}}{\sqrt{2}+y^2+\left\|x^3-\ln(y)\right\|}$	$x > 0$, $y > 0$
11	$l = 0,5x^5 + 3\cos(x+y) + e^{-0,1yz} - \sqrt{\left\|xy\right\|}$	z = 3.3, $x > 0$, $y > 0$

Окончание табл. 3.1

№ варианта	Выражения	Исходные данные
12	$g = \dfrac{1+\cos(x+y)}{\left\lvert e^{x} - 2y / \left(1 + x^{2}y^{2}\right)\right\rvert} x^{3} + \arcsin y$	$x > 0$, $y > 0$
13	$n = \sqrt{e^{x} + \operatorname{tg}\, x + 1}\left(\lg y + \cos xy + \sqrt[3]{x}\right)$	$x > 0$, $y > 0$
14	$a = \ln\left(y^{-\sqrt{\lvert x\rvert}}\right)\left(\sin x + e^{(x+y)}\right)$	$x > 0$, $y > 0$
15	$h = 2 + \dfrac{x^{2}}{\sqrt{2}} + \dfrac{\lvert y^{3}\rvert}{\sqrt{2}} + \dfrac{z^{4}(\ln x + 1)\sqrt{2}}{\sqrt{3}}$	$x > 0$, $y > 0$

Задание Б.

Разработать алгоритм и составить программу вычисления логического выражения по варианту задания из табл. 3.2. Исходные данные вводятся с клавиатуры по запросу программы, значения логических констант задаются присваиванием. На дисплей и в текстовый файл выводятся с указанием имен переменных значения исходных данных и результат вычислений.

Таблица 3.2

Варианты задания Б

№ вар.	Выражения	Исходные данные
1	x > 0 AND (y = 3 OR $x + y > 5$) OR $x - y < 0$	$x = 5$, $y = 2{,}5$
2	x >= 2 AND y <= 3 OR ($z = y$ AND $x + y + z > 0$)	$x = 7$, $y = -3{,}3$, $z = 0$
3	(a >= 5 AND b >= 5) AND ($a < 20$ AND $b < 30$)	$a = 20$, $b = 10$
4	($a = b$ OR $x + a < b$) AND ($x \cdot a <$ b OR $a > 10$)	$a = 15{,}5$, $b = 8{,}2$, $x = 7$

Окончание табл. 3.2

№ вар.	Выражения	Исходные данные
5	NOT a > 10 OR NOT (*b* <= 10 AND (c OR *b* >= 10))	*a* = 13, *b* = 20, *c* = TRUE
6	*x* + *y* > 3 OR y <= 7.3 OR *z* = *y* OR NOT (*x* > *z*)	*x* = 6, *y* = –6.5, *z* = 1.2
7	x <= 0 AND y >= 0 OR x < –5 AND y > 5.5	*x* = –2, *y* < 9
8	(NOT *xy* + *y* / *x* < 3,9 AND *xx* + *yy* >= 2.4)	*x* = 3, *y* = 4
9	NOT (*a* <= 0 OR *b* >= 0 AND (*p* OR *m*) AND c < 0)	*a* = 5, *b* = –3, *c* = –25.3, *p* = TRUE, *m* = FALSE
10	*a* < b AND *x* + *a* < *b* NOT (*p* OR *c* < *n*) OR *m*	*a* = 15,5, *b* = 8,2, *c* = –10,6, *n* = 0, *x* = 7, *p* = FALSE, *m* = FALSE
11	NOT (*x* > 2 AND *y* <=3 OR *z* <> *y* AND *x* + *z* > 0)	*x* = 10, *y* = –30, *z* = 1
12	NOT (*a* >= 15 OR NOT *b* < 30 AND *c* < 0 OR *b* <= 20)	*a* =5, *b* =13, *c* = –7
13	*a* + *x* >= 1 OR *c* + *d* < *a* NOT (*a* > *c* OR м) AND NOT *p*	*a* = 25,5, *d* = 18,2, *c* = –10,6, *m* = FALSE, *x* = 7, *p* = TRUE
14	*x* < 0 AND (*y* / *x* <= 0 OR *xy* >= 0) AND NOT *y* < 0	*x* = –7, *y* =2
15	NOT (*a* = 3 AND (*a* <> 5 OR NOT *a* + *x* < 1))	a = 4. x = 3

Задание В.

Ввести с клавиатуры произвольные значения вещественных переменных *p* и *t*. Используя эти значения, составить набор событий А и В на основе операций отношения (например, событие А как (*p* > *t*) и событие В – в виде (*t* <= 0)) для приведенных в табл. 3.3 вариантов соотношений результатов этих событий. Построить для логических операций **and, or** и **xor** логические выражения, соответствующие этим четырем вариантам, вычислить и вывести на дисплей и в текстовый файл их результаты с указанием названия логической операции. В отчете по лабораторной работе привести заполненную по результатам вычислений табл. 3.3.

Таблица 3.3

Истинность логических операций

Результаты операций отношения		Результаты логических операций		
Событие А	Событие В	A and B	A or B	A xor B
FALSE	FALSE			
FALSE	TRUE			
TRUE	FALSE			
TRUE	TRUE			

Содержание отчета

1. Цель работы.
2. Задание на лабораторную работу.
3. Блок-схема алгоритма.
4. Распечатка текста программы.
5. Распечатка результатов из текстового файла.
6. Истинность логических операций (только для задания В).
7. Выводы.

Лабораторная работа № 4

ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

Цель работы: получить навыки программирования алгоритмов разветвляющейся структуры.

Вводная информация

Понятие разветвляющегося алгоритма

Алгоритм называется *разветвляющимся*, если он содержит несколько отличающихся друг от друга вариантов последовательностей действий, а выбор одного из вариантов осуществляется в зависимости от выполнения или невыполнения какого-либо условия.

Типовые конструкции алгоритмического языка Турбо Паскаль для программирования разветвляющихся алгоритмов

Используются оператор перехода по условию **IF… THEN… ELSE** и обобщение оператора IF для произвольного числа вариантов – оператор выбора **CASE…OF**.

Структура *оператора перехода по условию* может быть представлена в двух вариантах:

IF Условие **THEN** Оператор;
или
IF Условие **THEN** Оператор1 **ELSE** Оператор2;

Здесь Условие – логическое значение TRUE или FALSE, представленное константой, переменной или логическим выражением либо результатом операции отношения:

```
b1:= TRUE; b2: = d<> c;
If b1 Then…       {условие – логическая переменная}
If not b2 Then… {условие – логическое выражение}
If x >= y Then…  {условие – результат операции отношения}
```

В первом варианте структуры, если Условие представлено значением TRUE, выполняется Оператор, записанный за словом THEN. Если Условие ложно, то выполняется следующий за оператором перехода по условию оператор программы.

Пример использования первого варианта оператора перехода по условию

```
x:= 1.1; y:= 2.2; z:= 0;

If x > y Then z:= x + y;{Условие ложно, оператор z:= x + y не выполнится}

z:= y / x; {Выполнится этот следующий оператор программы}
writeln ('z =', z);
```

Во втором варианте при значении Условия TRUE выполняется Оператор1, при значении Условия FALSE – Оператор2.

Пример использования второго варианта оператора перехода по условию

```
x:= 1.1; y:= 2.2; z:= 0;

If x > y Then writeln ('x больше y')
         Else writeln ('x меньше или равен y');
```

Примечание. Операторы, стоящие после **Then** или **Else,** могут быть простыми или составными. Составной оператор – последовательность из нескольких операторов, заключенных в операторные скобки **begin – end**.

Операторы перехода по условию могут быть вложенными друг в друга:

```
If Условие1
    THEN          {Условие1 истинно}
      If Условие2
         then     {Условие2 истинно}
           begin
            . . . . . {составной оператор 1}
              end
```

```
    else   {Условие2 ложно}
      простой оператор
  ELSE     {Условие1 ложно}
    begin
    . . . . .  {составной оператор 2}
    end;
```

При вложенности операторов каждое **else** соответствует тому **then**, которое непосредственно ему предшествует.

Оператор выбора **case** обеспечивает выбор одного из произвольного числа вариантов. Он содержит селектор в виде *Управляющей_переменной* или *Выражения* и набор операторов, каждому из которых предшествует список констант выбора. Список может состоять из одного или нескольких значений и (или) диапазонов значений (границы диапазонов записываются через символ диапазона «...»), разделенных запятыми. Тип констант выбора должен быть совместим с типом селектора.

Структура оператора выбора:

```
  case селектор of
  список1: оператор1;
  список2: оператор2;
  . . . . . .
  список N: операторN
    else оператор N + 1 {необязательный элемент оператора выбора}
end; {case }
```

При передаче управления оператору выбора сначала определяется значение селектора, затем выполняется тот оператор, константа выбора из списка которого равна текущему значению селектора. Оператор может быть составным.

Если текущее значение селектора не равно ни одной из констант списков выбора, выполняется оператор, следующий за служебным словом **else**. Если слово **else** не используется в структуре оператора, управление передается следующему оператору за границей **case** (то есть за словом **end**).

Не допускается в качестве селектора использовать вещественный и строковый типы.

Примеры использования оператора выбора:

```
{селектор – Выражение k+1}
Var k, r: byte;
BEGIN
Case k + 1 of
1, 2, 6, 14, 22, 43, 47: r:= k + 5; {константы выбора – список значений}
3..5, 7..13, 55..65: r:= k * 2; {константы выбора – список интервалов}
16..21, 23, 27..40:  r:= k – 2  {константы выбора – значения и интервалы}
end;
END.
Var{селектор – Управляющая_переменная перечислимого типа}
Season: (Zima, Vesna, Leto, Osen);
BEGIN
Case Season of
  Zima: writeln (‘Холодно’);
  Vesna: writeln (‘Потеплело’);
  Leto: writeln (‘Очень тепло’);
  Osen: writeln (‘Похолодало’)
    Else writeln (‘Мы на Марсе’)
End; {case}
END.
```

Задания на лабораторную работу

Задание А.

Разработать алгоритм и составить программу вычисления выражения из табл. 4.1. Исходные данные вещественного типа вводятся с клавиатуры по запросу программы. На дисплей и в текстовый файл выводятся значения исходных данных и результаты вычислений.

Таблица 4.1

Варианты задания А

№ варианта	Выражения	Переменные выражений
1	$y=\begin{cases}\dfrac{2,3x+3,56}{\operatorname{tg}(x+1)}, & x<3\\ \sin(5,4x^2-x^{2-x}), & 3\le x\le 8\\ \operatorname{tg}(x+3)-\dfrac{1}{x}, & x>8\end{cases}$	x
2	$y=\begin{cases}\sin(x+2), & x<2\\ x+3,5\operatorname{tg}x, & 2\le x\le 4\\ \sqrt{\lvert 2,56x-0,35\rvert}, & x>4\end{cases}$	x
3	$y=\begin{cases}\ln(x+2), & -2<x\le 0\\ \sqrt{2-x}, & x\le -2\\ e^x+\sqrt{x+4}, & x>0\end{cases}$	x
4	$y=\begin{cases}\operatorname{tg}(\lvert 2x+4,2\rvert)-\lg\lvert x\rvert, & x<2\\ \sin x+\sqrt{6x}, & 2\le x\le 7\\ 6+\operatorname{arctg}\left(\dfrac{2x}{1+\sqrt{x}}\right), & x>7\end{cases}$	x
5	$y=\begin{cases}6,25+7x, & x<6\\ 5\ln(\lvert 2-x^2\rvert), & 6\le x\le 8\\ \dfrac{25x}{\operatorname{tg}x}, & x>8\end{cases}$	x

Продолжение табл. 4.1

№ варианта	Выражения	Переменные выражений
6	$y=\begin{cases}5\sqrt{x+2}, & 0<x\le 1\\ \ln(x-1), & x>1\\ e^{-x}, & x\le 0\end{cases}$	x
7	$y=\begin{cases}e^{2x+1}, & 0<x<1\\ \sqrt{1-x}, & x\le 0\\ \ln x, & x\ge 1\end{cases}$	x
8	$y=\begin{cases}5\sqrt{x+2}, & 0<x\le 1\\ \ln(x-1), & x>1\\ e^{-x}, & x\le 0\end{cases}$	x
9	$y=\begin{cases}\operatorname{tg}(2x+4,2)-2x, & x<2\\ \sin x+\sqrt{6x}, & 2\le x\le 5\\ 3,56x+\dfrac{2+x}{1+\sqrt{x}}, & x>5\end{cases}$	x
10	$y=\begin{cases}\ln(\lvert 5,3x^{3x}-x^2\rvert), & x<1\\ \dfrac{0,025}{\operatorname{tg}(2,6+x)}, & 1\le x\le 4\\ \sin^2(x-6), & x>4\end{cases}$	x
11	$y=\begin{cases}\dfrac{2,3x+3,56}{\operatorname{tg}(x+1)}, & x<3\\ \sin(5,4x^2-x^{2-x}), & 3\le x\le 8\\ \operatorname{tg}(x+3)-\dfrac{1}{x}, & x>8\end{cases}$	x

Окончание табл. 4.1

№ варианта	Выражения	Переменные выражений
12	$n=\begin{cases}3k^2+2\mathrm{m}^3, & k>\lvert m\rvert \\ \lvert k-m\rvert, & 3<k<\lvert m\rvert \\ (k-m)^3, & k=\lvert m\rvert\end{cases}$	*k, m*
13	$d=\begin{cases}\ln(\lvert\mathrm{f}\rvert+\lvert\mathrm{q}\rvert, & \lvert fq\rvert>10 \\ e^{f+q}, & \lvert fq\rvert<10 \\ f+q, & \lvert fq\rvert=10\end{cases}$	*f, q*
14	$z=\begin{cases}(\mathrm{x}-\mathrm{y})^3+\mathrm{arctg}(\mathrm{x}), & x>y \\ (y-x)^3+\mathrm{arctg}(\mathrm{x}), & x<y \\ (y-x)^3+1, & x=y\end{cases}$	*x, y*
15	$y=\begin{cases}b\cos\left(\dfrac{\dfrac{x^3}{b}+1}{x}\right), & \sin\left(b+\dfrac{1}{x}\right)<0 \\ x\sin\left(b+\dfrac{1}{x}\right), & \sin\left(b+\dfrac{1}{x}\right)<0\end{cases}$	$b=0{,}5$ x

Задание Б.

Разработать алгоритм и составить программу вычисления выражения из табл. 4.1 с использованием оператора выбора **case**. Исходные данные вещественного типа вводятся с клавиатуры по запросу программы. На дисплей и в текстовый файл выводятся значения исходных данных и результаты вычислений. Протестировать все ветви алгоритма.

Примечание. Так как селектор не может быть вещественного типа, следует поставить в соответствие каждому условию значение константы, тип которой может использоваться в качестве селектора оператора выбора **case**.

Задание В.

Разработать алгоритм и составить программу решения задачи для указанного преподавателем варианта из табл. 4.2.

Таблица 4.2

Варианты задания В

№ варианта	Задачи
1	Программа должна запрашивать ввод целого числа в диапазоне от 1 до 7, обозначающего порядковый номер дня недели, выводить на дисплей и в текстовый файл введенное число и соответствующее введенному номеру название этого дня (1 – понедельник, 2 – вторник, ..., 7 – воскресенье) и одно из сообщений: «Рабочий день», «Суббота», «Воскресенье». При вводе недопустимого значения программа должна вывести сообщение «Число вне диапазона»
2	Программа должна запрашивать ввод натурального числа N ($N <= 10$), определяющего сумму денег в рублях, выводить на дисплей и в текстовый файл введенное число и соответствующее слово: «рубль», «рубля» или «рублей». При вводе недопустимого значения программа должна вывести сообщение «Число вне диапазона» и повторить запрос на ввод числа
3	Программа должна запрашивать ввод целого числа в диапазоне от 1 до 10, выводить на дисплей и в текстовый файл введенное число и название этого числа (1 – один, 2 – два, …, 10 – десять). При вводе недопустимого значения программа должна вывести сообщение «Число вне диапазона» и повторить запрос на ввод числа
4	Программа должна запрашивать ввод целого числа в диапазоне от 1 до 12, определяющего номер соответствующего месяца, выводить на дисплей и в текстовый файл введенное число, название этого месяца и поры года, соответствующее выбранному месяцу. При вводе недопустимого значения числа программа должна вывести сообщение «Ошибка ввода»
5	Программа должна запрашивать ввод целого числа в диапазоне от 1 до 12, определяющего номер соответствующего месяца, выводить на дисплей и в текстовый файл введенное число, а также название этого месяца года (1 – январь, 2 – февраль, ..., 12 – декабрь) и количество дней в нем. При вводе недопустимого значения программа должна вывести сообщение «Число вне диапазона» и повторить запрос на ввод числа

Продолжение табл. 4.2

№ варианта	Задачи
6	Программа должна запрашивать ввод целого натурального числа N ($N < 100$), определяющего возраст человека в годах, выводить на дисплей и в текстовый файл введенное число с добавлением соответствующего этому числу варианта наименования «год», «года», «лет». При вводе недопустимого значения программа должна вывести сообщение «Ошибка ввода»
7	Программа должна запрашивать ввод возраста ребенка в диапазоне 6..18 лет и определять его в соответствующий класс средней школы: с 6 до 7 лет – 1-й класс, с 7 до 8 лет – 2-й, с 8 до 9 лет – 3-й, с 9 до 10 лет – 4-й и т. д. На дисплей и в текстовый файл выводятся сообщение о возрасте ребенка и рекомендуемом классе для обучения. При вводе недопустимого значения программа должна вывести сообщение «Ошибка ввода» и повторить запрос на ввод.
8	Программа должна запрашивать ввод целого числа в диапазоне от –10 до 10, выводить на дисплей и в текстовый файл введенное число, а также сообщение «Введенное число больше 0 (меньше 0 или равно 0)». При вводе недопустимого значения числа программа выводится сообщение «Число вне диапазона» и повторить запрос на ввод числа
9	Программа должна запрашивать ввод целого числа в диапазоне от 2 до 22, выводить на дисплей и в текстовый файл введенное число и сообщение «Введенное число четное (нечетное)». При вводе недопустимого значения числа программа выводится сообщение «Число вне диапазона» и повторить запрос на ввод числа
10	Программа должна запрашивать ввод значения текущего года и года поступления в университет, выводить на дисплей и в текстовый файл введенные числа и название курса (первый, второй и т. д.), на котором обучается студент
11	Программа должна запрашивать ввод значения текущего времени в часах (без минут) в диапазоне от 0 до 24, выводить на дисплей и в текстовый файл введенное значение и соответствующее сообщение: «Это утро», «Это день», «Это вечер», «Это ночь». При вводе недопустимого значения числа программа должна вывести сообщение «Ошибка ввода» и повторить запрос на ввод

Окончание табл. 4.2

№ варианта	Задачи
12	Программа должна запрашивать ввод значения года в укороченной форме (от 0 по 10), выводить на дисплей и в текстовый файл введенное значение и год текущего столетия в полном формате (0–2000, 1–2001 и т. д.). При вводе недопустимого значения числа программа должна вывести сообщение «Ошибка ввода» и повторить запрос на ввод
13	Программа должна запрашивать ввод оценки в диапазоне от 1 до 10, полученной студентом на экзамене и выводить на дисплей и в текстовый файл полученную оценку и одно из сообщений «очень плохо», «плохо», «удовлетворительно», «хорошо», «отлично». При вводе числа вне диапазона выводится соответствующее сообщение и повторяется запрос на ввод числа
14	Программа должна запрашивать ввод символа путем нажатия любой клавиши на клавиатуре, анализировать введенный символ и выводить на дисплей и в текстовый файл сообщение вида: «Это буква латинского алфавита» или «Это буква русского алфавита», или «Это цифра», или «Это специальный символ» и значение введенного символа
15	Программа должна выводить на дисплей подсказку о буквенных обозначениях четырех стандартных функций (a – abs(x), s – sin(x), c – cos(x), l – ln(x)), затем запрашивать у пользователя ввод целого числа $x > 1$ и символа, обозначающего функцию, вычислять значение выбранной функции от x, выводить на дисплей и в текстовый файл введенное число и значение функции с указанием ее имени

Содержание отчета

1. Цель работы.
2. Задание на лабораторную работу.
3. Блок-схема алгоритма.
4. Распечатка текста программы.
5. Распечатка результатов из текстового файла.
6. Выводы.

Лабораторная работа № 5

ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Цель работы: практически освоить программирование циклических вычислительных процессов.

Вводная информация

Понятие циклического алгоритма

Алгоритм называется *циклическим*, если при его исполнении некоторые операторы неоднократно повторяются с новыми значениями переменной цикла, не смотря на то, что записаны они в тексте программы один раз.

Средства языка Турбо Паскаль для организации циклических вычислений

В Турбо Паскале реализуются следующие циклические структуры:

– *цикл с предусловием* (цикл «ПОКА») – условие расположено в начале циклической структуры;

– *цикл с постусловием* (цикл «ДО») – условие расположено в конце циклической структуры;

– цикл с параметром.

Если количество повторов заранее известно, то используется цикл с параметром, если количество повторов неизвестно, применяются цикл с постусловием или цикл с предусловием.

Оператор *цикла с предусловием* имеет следующий формат записи:

WHILE условие **DO** тело цикла;

Условие представляет собой логическое выражение, а тело цикла – простой или составной оператор. Значение условия вычисляется перед каждым выполнением тела цикла. Пока значение условия *True,* выполняются операторы тела цикла и снова вычисляется условие. Если значение условия становится *False*, происходит

выход из цикла и передача управления следующему за границей цикла оператору программы.

Если первое значение условия равно *False,* операторы тела цикла не выполняются ни разу.

Все операнды, входящие в условие повторения цикла, должны быть определены до обращения к оператору цикла.

В теле цикла должно быть хотя бы одно действие, приводящее к изменению хотя бы одной переменной, входящей в логическое выражение условия, иначе цикл будет бесконечным.

Использование цикла с предусловием рассмотрим на следующем примере. Требуется составить программу для вывода на дисплей таблицы умножения целых чисел в диапазоне от 2 до 9 в виде axb = c. Число, для которого следует вывести таблицу, вводится с клавиатуры по запросу программы.

Программа имеет вид:

```
Program tabl1;
Label m1;
Var
i, j, k, p: byte;
BEGIN
Write (‘Таблицу какого числа вывести (целое в диапазоне от 2 до 9)?’);
Readln(i);
J:= 2;
While j <= 9 do   {проверка условия}
      begin
      k:= i * j;
      Writeln (i: 2,’x’, j: 2, ‘=’, k: 2;
      j:= j + 1;   {изменение переменной, влияющей на условие}
      end; {while}
Write (‘Будем продолжать (да – 1, нет – 0)’);
Readln (p);
If p = 1 goto m1;
END.
```

Оператор *цикла с постусловием* имеет следующий формат записи:

REPEAT тело цикла **UNTIL** условие;

Тело цикла может состоять из любого количества любых операторов, расположенных между **REPEAT** и **UNTIL**, то есть операторы **REPEAT** и **UNTIL** играют роль операторных скобок **begin – end** для составного оператора тела цикла.

Значение результата вычисления условия, проверяется после каждого выполнения тела цикла, то есть при любом значении условия операторы тела цикла выполнятся хотя бы один раз.

Если результат вычисления условия FALSE, то повторяется выполнение операторов тела цикла; если результат TRUE, то работа цикла прекращается и выполняется следующий оператор, стоящий за границей оператора цикла (за условием).

Используемые в логическом условии операнды должны быть определены до вычисления условия.

В теле цикла должно быть хотя бы одно действие, приводящее к изменению хотя бы одной переменной, входящей в условие окончания цикла, так как в противном случае цикл будет бесконечным.

Программа для предыдущего примера с использованием оператора цикла с постусловием:

```
Program tabl2;
Label m1;
Var
i, j, k, p: byte;
BEGIN
m1: Write (‘Таблицу умножения какого целого числа’);
Write (‘в диапазоне от 2 до 9 вывести на дисплей ?’);
Readln(i);
J:= 2;
Repeat
k:= i * j;
writeln (i: 2,’x’, j: 2, ‘=’, k: 2;
j:= j + 1;   {изменение переменной, влияющей на условие}
until j > 9;  {проверка условия}
```

```
write (‘Будем продолжать (да – 1, нет – 0’);
Readln (p);
If p = 1 goto m1;
END.
```

Оператор *цикла с параметром* используется в случаях, когда значения переменной изменяются на единицу и для каждого значения переменной из диапазона ее изменения необходимо выполнять одинаковые действия. Эта переменная называется параметром цикла.

Форматы записи оператора:

FOR параметр_ цикла:= S1 **TO** S2 **DO** оператор; (вариант 1)
или
FOR параметр_цикла:= S2 **DOWNTO** S1 **DO** оператор; (вариант 2)

Здесь S1 и S2 – соответственно начальное и конечное значения параметра цикла, а оператор (простой или составной) – тело цикла.

Переменная цикла, ее начальное и конечное значения должны быть одного типа. Допустимые типы – скалярные, кроме вещественного (целые, символьные, логические), ограниченный (интервальный) и перечислимый.

Если используются типы **byte**, **integer** и интервальный, то в первом варианте формата записи переменная цикла принимает последовательно возрастающие на единицу значения от S1 до S2, а во втором варианте переменная цикла последовательно убывает на единицу от значения S2 до S1. Количество повторов фиксирует встроенный внутренний счетчик цикла. Если S1 = S2, цикл выполнится один раз. Если S1 > S2 при возрастающем параметре цикла (**FOR… TO**) или S2 < S1 при убывающем параметре цикла (**FOR… DOWNTO**), цикл не выполнится ни разу.

Программа для предыдущего примера с использованием оператора цикла с параметром:

```
Program tabl3;
Label m1;
Var
i, j, k, p: byte;
```

```
BEGIN
m1: Write (‘Таблицу умножения какого целого числа’);
Write (‘в диапазоне от 2 до 9 вывести на дисплей?’);
Readln (i);
For j:= 2 to 9 do
    begin
    k:= i * j;
    Writeln (i: 2,’x’, j: 2, ‘=’, k: 2;
    end;
Write (‘Будем продолжать? (да – 1, нет – 0)’);
Readln (p);
If p = 1 goto m1;
END.
```

Во всех трех операторах цикла можно реализовать досрочный выход из цикла с помощью оператора безусловного перехода **goto** при выполнении в теле цикла определенного логического условия в операторе условного перехода **if**… **then**. Но передавать управление извне цикла внутрь цикла нельзя.

Все рассмотренные операторы цикла могут использоваться в теле другого цикла (вложенные циклы). Границы внутреннего вложенного цикла не должны выходить за границы внешнего цикла:

```
For i: = 1 to N do        {цикл по i – внешний}
    Begin
Оператор 1;
Оператор 2;
For j:= 1 to M do         {цикл по j – вложенный}
    Begin
    Операторы тела цикла по j
        End {цикла по j}
        End; {цикла по i}
```

При передаче управления вложенному циклу от внешнего возврат к внешнему циклу возможен, когда вложенный цикл отработает полностью, то есть когда выполнится условие завершения его работы.

Задания на лабораторную работу

Задание А.

Разработать алгоритм и составить программу табулирования функции по одному из вариантов (табл. 5.1). Вычисления значений функции выполнить циклом с предусловием. Начальное и конечные значения аргумента и шаг его приращения вводятся с клавиатуры по запросу программы. Таблица должна иметь шапку вида:

```
-----------------------------------
I Аргумент I Функция I
-----------------------------------
```

В качестве разделителя строк использовать линию, образованную символами «-».

На дисплей и в текстовый файл выводятся исходные данные с указанием имен переменных и результаты работы программы в виде таблицы.

Таблица 5.1

Варианты задания А

№ варианта	Функция	Диапазон и шаг изменения аргумента
1	$F(x) = \sin x + \cos x$	$x \in [-\pi; \pi]$ $dx = \pi/6$
2	$F(x) = \ln x - \frac{1}{x+5}$	$x \in [1{,}8; 2{,}8]$ $dx = 0{,}1$
3	$F(x) = 1 + \cos(2x - 3)$	$x \in [-\pi; \pi]$ $dx = \pi/6$
4	$F(x) = 2 + \cos(x / 2)$	$x \in [-\pi; \pi]$ $dx = \pi/6$
5	$F(x) = \frac{\sin x}{e^x}$	$x \in [-\pi; \pi]$ $dx = \pi/6$

Продолжение табл. 5.1

№ варианта	Функция	Диапазон и шаг изменения аргумента
6	$F(x)=\frac{\cos^2 x}{x+1}$	$x \in [-\pi; \pi]$ $dx = \pi/6$
7	$F(x)=e^x-\frac{1}{x+2}$	$x \in [-1{,}2; 0{,}6]$ $dx = 0{,}2$
8	$F(x)=e^x-\frac{1}{e^x}$	$x \in [-1{,}8; 1{,}6]$ $dx = 0{,}2$
9	$F(x)=\left(\sin x+\cos x\right)^2$	$x \in [-\pi/2; \pi/2]$ $dx = \pi/6$
10	$F(x)=e^x-\frac{1}{x+2}$	$x \in [-1{,}5; 1{,}5]$ $dx = 0{,}25$
11	$F(x)=\sqrt[5]{5^x}+\frac{x^3}{\log_8(1+x^2)}$	$x \in [0{,}2; 2{,}2]$ $dx = 0{,}2$
12	$F(x)=\frac{x}{5}\log_5(x^2+\sqrt{x^2+1})$	$x \in [-1{,}4; 1{,}8]$ $dx = 0{,}2$
13	$F(x)=\frac{3}{\sqrt{x^2+1}}+\lg^2(x+2)$	$x \in [-0{,}4; 1{,}2]$ $dx = 0{,}2$
14	$F(x)-\frac{2}{x^2}\cos\frac{1}{x}$	$x \in [-\pi; \pi]$ $dx = \pi/6$
15	$F(x)=\frac{x^3}{\sqrt{7+x^2}}$	$x \in [-1{,}8; 1{,}6]$ $dx = 0{,}2$

Задание Б.

Разработать алгоритм и составить программу вычисления циклом с постусловием бесконечной суммы по варианту задания из табл. 5.2. Начальное значение переменной, шаг ее приращения и точность вычисления суммы вводятся с клавиатуры по запросу

программы. Промежуточные значения переменной и слагаемых выводятся на дисплей по формату с фиксированной десятичной точкой. Накопление суммы прекращается, когда значение последнего вычисленного слагаемого становится соизмеримым по абсолютной величине с заданной точностью. Программа должна зафиксировать, сколько раз выполнился цикл до достижения заданной точности вычислений. Значение последнего слагаемого и заданной точности вычислений, а также значение суммы и количество выполненных повторений вывести на дисплей и в текстовый файл.

Таблица 5.2

Варианты задания Б

№ варианта	Выражение	Исходные данные	Точность вычислений
1	$S = \sum_x \frac{2x+2}{3x^3+x^2}$	$x_{нач} = 0{,}5$ $dx = 0{,}5$	$\varepsilon = 10^{-3}$
2	$S = \sum_x \frac{3x^2+2}{5e^x+x}$	$x_{нач} = 0{,}5$ $dx = 0{,}5$	$\varepsilon = 10^{-3}$
3	$S = \sum_x \frac{2x^2+x}{3x^3+4x^2}$	$x_{нач} = 1$ $dx = 0{,}5$	$\varepsilon = 10^{-3}$
4	$S = \sum_x \frac{x^2+2x}{e^x+3\frac{x^3}{2}}$	$x_{нач} = 0{,}5$ $dx = 0{,}25$	$\varepsilon = 10^{-3}$
5	$S = \sum_x \frac{x+e^{-x}}{x^3+e^x}$	$x_{нач} = 0{,}5$ $dx = 0{,}25$	$\varepsilon = 10^{-4}$
6	$S = \sum_x \frac{2x+1}{x^x+\frac{x}{3}}$	$x_{нач} = 0{,}5$ $dx = 0{,}5$	$\varepsilon = 10^{-3}$
7	$S = \sum_n (-1)^n \frac{n+2}{n^n+1}$	$n_{нач} = 1$ $dn = 1$	$\varepsilon = 10^{-4}$

Окончание табл. 5.2

№ варианта	Выражение	Исходные данные	Точность вычислений
8	$S=\sum_n (-1)^n \dfrac{\sqrt{n}+\dfrac{n}{2}}{n^n}$	$n_{\text{нач}}=1$ $\mathrm{d}n=1$	$\varepsilon=10^{-3}$
9	$S=\sum_n (-1)^n \dfrac{n^2+\dfrac{5}{n}}{n^n}$	$n_{\text{нач}}=1$ $\mathrm{d}n=1$	$\varepsilon=10^{-4}$
10	$S=\sum_n (-1)^n \dfrac{n^{n-1}+\dfrac{n}{2}}{\dfrac{n^n}{2}+2n}$	$n_{\text{нач}}=2$ $\mathrm{d}n=1$	$\varepsilon=10^{-3}$
11	$S=\sum_n (-1)^n \left(\dfrac{k}{(n+1)(n+2)}\right)^n$	$k=2$ $n_{\text{нач}}=1$ $\mathrm{d}n=1$	$\varepsilon=10^{-3}$
12	$S=\sum_n (-1)^n \left(\dfrac{k}{(n+1)(n+2)(n+3)}\right)^{2n}$	$k=3$ $n_{\text{нач}}=1$ $\mathrm{d}n=1$	$\varepsilon=10^{-3}$
13	$S=\sum_n (-1)^n \dfrac{k^n}{k^{2n}+2}$	$k=2$ $n_{\text{нач}}=1$ $\mathrm{d}n=1$	$\varepsilon=10^{-3}$
14	$S=\sum_n (-1)^{n+1} \dfrac{2n+3}{5n^3+n^2}$	$n_{\text{нач}}=1$ $\mathrm{d}n=1$	$\varepsilon=10^{-3}$
15	$S=\sum_n (-1)^{n-1} \dfrac{5n^2-n}{4n^3+3n^2}$	$n_{\text{нач}}=2$ $\mathrm{d}n=1$	$\varepsilon=10^{-3}$

Задание В.

Разработать алгоритм и составить программу вычисления циклом с параметром суммы первых n членов степенного ряда по варианту задания из табл. 5.3. Значения переменной x и количество n членов степенного ряда вводятся с клавиатуры. Начальное значение $n_{\text{нач}} = 1$, шаг приращения $\mathrm{d}n = 1$. Если в степенном ряде исполь-

зуется факториал, то вычисление факториала также выполнить с использованием цикла с параметром.

Таблица 5.3

Варианты задания В

№ варианта	Степенной ряд
1	$y = 1 + \frac{\cos x}{1!} + \frac{\cos 2x}{2!} + \cdots + \frac{\cos nx}{n!}$
2	$y = 1 + \frac{x^2}{2!} - \frac{3x^4}{4!} + \cdots + (-1)^n \frac{2n-1}{(2n)!} x^{2n}$
3	$y = \frac{x^3}{5} - \frac{x^5}{17} + \cdots + (-1)^{n+1} \frac{x^{2n+1}}{4n^2+1}$
4	$y = 1 + \frac{\cos(\pi/4)}{1!} x + \cdots + \frac{\cos(n\pi/4)}{n!} x^n$
5	$y = \frac{1}{x} - \frac{1}{3x^3} + \frac{1}{5x^5} + \cdots + (1)^n \frac{1}{(2n+1)x^{2n+1}}$
6	$y = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^n + 1 \frac{x^{2n-1}}{(2n-1)!}$
7	$y = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots + (-1)^n \frac{x^{2n}}{(2n)!}$
8	$y = x - \frac{x^2}{1-2} + \frac{x^3}{2-4} - \frac{x^4}{3-8} + \cdots + (-1)^{n+1} \frac{x^{n+1}}{n-2^n}$
9	$y = \frac{x\ln x}{1!} + \frac{(x\ln x)^2}{2!} + \cdots + \frac{(x\ln x)^n}{n!}$
10	$y = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots + (-1)^{n+1} \frac{x^n}{n}$
11	$y = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots - \frac{x^n}{n}$

Окончание табл. 5.3

№ варианта	Степенной ряд
12	$y = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + \frac{x^{2n+1}}{(2n+1)!}$
13	$y = x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots + (-1)^{n+1}\frac{x^{2n+1}}{(2n+1)!}$
14	$y = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^{n}\frac{x^{n+2}}{(2n+1)!}$
15	$y = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} + \cdots + (-1)^{n+1}\frac{(x-1)^n}{n}$

Содержание отчета

1. Цель работы.
2. Задание на лабораторную работу.
3. Блок-схема алгоритма.
4. Распечатка текста программы.
5. Распечатка результатов из текстового файла.
6. Выводы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Офицеров, Д. В. Программирование на персональных ЭВМ. Практикум: учебное пособие / Д. В. Офицеров, А. Б. Долгий, В. А. Старых. – Минск: Выш. шк., 1993. – 256 с.

2. Бармин, В. А. Вычислительная техника и программирование: в 2 ч. / В. А. Бармин, А. П. Киселева, В. В. Трикозенко. – Минск: БНТУ, 1998. – Ч. 1: Программирование в среде TURBO PASCAL 7.0. – 116 с.

3. Лабораторный практикум по программированию на языке Паскаль: учебное пособие. / Л. В. Найханова [и др.]; под общ. ред. Л. В. Найхановой, Н. Ц. Бильгаевой. – 3-е изд. доп. и перераб. – Улан-Удэ, 2004. – 176 с.

4. Фаронов, В. В. Программирование на персональных ЭВМ в среде Турбо–Паскаль / В. В. Фаронов. – Москва: МГТУ, 1990. – 580 с.

СОДЕРЖАНИЕ

Учебное издание

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

Лабораторный практикум

В 3 частях

Часть 2

ПРОГРАММИРОВАНИЕ ТИПОВЫХ АЛГОРИТМОВ

Составитель
ПЕТРЕНКО Станислав Михайлович

Редактор *Е. С. Кочерго*
Компьютерная верстка *Е. А. Беспанской*

Подписано в печать 28.02.2018. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 2,73. Уч.-изд. л. 2,14. Тираж 100. Заказ 587.

Издатель и полиграфическое исполнение: Белорусский национальный технический университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий № 1/173 от 12.02.2014. Пр. Независимости, 65. 220013, г. Минск.