Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Орский гуманитарно-технологический институт (филиал)
Государственного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»

А. Н. ШАКАЛОВ

Турбо Паскаль 7.0

Утверждено редакционно-издательским советом ОГТИ
в качестве лабораторного практикума

Орск 2008

Содержание

ВВЕДЕНИЕ

Настоящий лабораторный практикум предназначен для использования в курсе «Информатика», «Вычислительный практикум на ЭВМ» для всех специальностей высшего учебного заведения. Практикум позволяет построить курс в соответствии с разнообразным планированием и является гибким инструментом обучения алгоритмическому мышлению. В начале текста лабораторных работ указывается максимальное время, которое рекомендуется затратить на ее выполнение. Практикум направлен на формирование навыков реализации алгоритмов на популярном языке высокого уровня Турбо Паскаль. В каждой работе практикума содержится, или генерируется, по 30 вариантов заданий, которые мало различаются по уровню необходимых навыков и по сложности выполнения. Подготовка к выполнению заданий достигается наличием в каждой работе разработанного типового примера этого раздела программирования. При выполнении в полном объеме практикум рассчитан на 34 часа и содержит десять работ:

1. Введение в Турбо Паскаль (2 часа).
2. Переменные (2 часа).
3. Графика (6 часов).
4. Циклы (6 часов).
5. Условный переход (2 часа).
6. Массивы (6 часов).
7. Подпрограммы (2 часа).
8. Работа со строками (4 часа).
9. Файлы (2 часа).
10. Запись и множество (2 часа).

Практикум выполняется дифференцированно (по объему) в курсах «Информатика» и «Вычислительный практикум на ЭВМ» в соответствии с программами и методическими материалами. Пособие предназначено для обучения основам алгоритмизации и программирования и выполнения на языке Турбо Паскаль, однако тексты задач не подразумевают решения на каком-то конкретном языке программирования, и алгоритмы могут быть реализованы на любом языке программирования высокого уровня.

Практикум выполняется на персональных компьютерах IBM и совместимых с ними. При выполнении студент по каждой работе оформляет отчет по требованиям, сформулированным во вступительной части лабораторной работы. Требования к оформлению отчетов едины, но в зависимости от материала меняется их комплектация.

ЛАБОРАТОРНАЯ РАБОТА 1.
ВВЕДЕНИЕ В ТУРБО ПАСКАЛЬ (TURBO PASCAL 7.0) ЗНАКОМСТВО С ИНТЕГРИРОВАННОЙ СРЕДОЙ. ИСПОЛЬЗОВАНИЕ КОМПЬЮТЕРА ДЛЯ ВЫЧИСЛЕНИЯ ЗНАЧЕНИЙ ВЫРАЖЕНИЙ (РЕЖИМ МИКРОКАЛЬКУЛЯТОРА) (2 ЧАСА)

Цель работы: приобретение навыков работы с интегрированной средой Турбо Паскаль. Запуск среды Турбо Паскаль. Освоение правил построения выражений, копирование строк, сохранение программ. Выход из среды Турбо Паскаль.

Теоретические положения
Интегрированная среда Турбо Паскаль

Для общения с компьютером нужен язык. В дальнейшем будет использоваться один из вариантов языка программирования PASCAL, созданный на базе английского языка (язык назван в честь знаменитого ученого Блеза Паскаля).

Паскаль [Pascal] – процедурно-ориентированный язык программирования высокого уровня, предназначенный для решения вычислительных и информационно-логических задач.

Первоначальная ориентация языка на обучение структурному программированию. Разработан в 1968-1971 гг. Никлаусом Виртом в Цюрихском Институте информатики (Швейцарии). В настоящее время Паскаль принадлежит к группе наиболее распространенных и популярных языков программирования. Существуют многочисленные реализации практически для всех машинных архитектур.

Турбо Паскаль 1.0 был написан датчанином Андресом Хейльсбергом и продавался в Европе под названием Kompass Pascal. Филипп Канн (США) приобрел его и создал компанию Borland International на основе концепции продажи компилятора непосредственно пользователям. Изучаемая в данном практикуме версия – Турбо Паскаль 7.0.

Турбо Паскаль – это, вообще говоря, не просто язык, а некоторая система или среда, которая, кроме собственно языка программи-

рования PASCAL, содержит универсальный текстовый многооконный редактор, компилятор входного языка, редактор связей, встроенный отладчик, развитую систему меню, что обеспечивает высокую производительность труда программиста. Таким образом, Турбо Паскаль является расширением оригинальной версии языка PASCAL.

В дисковой операционной системе почти одновременно с Турбо Паскалем появилась версия Борланд Паскаль, которая по возможностям совершенно сравнима с ним, а по качеству распределения памяти компьютера его превосходит.

Для управления средой Турбо Паскаля используются функциональные клавиши F1 – F12. С каждой из этих клавиш связывается некоторая команда, управляющая средой. Кроме того, допускается модификация этих клавиш с помощью клавиш Alt, Ctrl и Shift.

Роль этих клавиш будет раскрываться по мере необходимости.

Остановимся на описании некоторых их действий.

Таблица 1

Функциональные клавиши	**Действия функциональных клавиш в среде ТП7**
1	2
F1	Обратиться к справке
F2	Сохранить текущее окно в дисковый файл
F3	Загрузить в интегрированную среду дисковый файл
F4	Используется в отладочном режиме: начать или продолжить исполнение программы и остановиться перед исполнением той ее строки, на которой стоит курсор
F5	Распахнуть активное окно на весь экран или вернуть прежние размеры
F6	Активизировать следующее по номеру окно редактора
F7	Пройти программу по шагам, с пошаговым выполнением подпрограммы
F8	Пройти программу по шагам с быстрым выполнением подпрограммы
F10	Выйти в главное меню
Ctrl – F1	Вызвать справочную информацию об операторе программы, на котором стоит курсор, если курсор не находится на зарезервированном слове, то будет выдана индексная страница слов, начало которых совпадает с искомым

Окончание табл. 1

1	2
Ctrl – F9	Прогон программы: компилировать программу, находящуюся в редакторе, загрузить ее в оперативную память, выполнить и вернуться в среду Турбо Паскаля
Alt – F3	Закрыть активное окно
Alt – F9	Компилировать (без выполнения) программу
Alt – F5	Сменить окно редактирования на окно вывода результатов прогонки программы (для возврата в режим редактирования достаточно нажать любую кнопку)
Alt – X	Выйти из среды Турбо Паскаля

Режим редактирования устанавливается сразу после загрузки Турбо Паскаля. Признаком готовности является появление курсора в верхней части активного окна редактора.

Для загрузки Турбо Паскаля сначала следует войти в интегрированную среду Турбо Паскаль. Для этого нужно запустить файл TURBO.EXE. Он при стандартных настройках располагается в каталоге TP\BIN. Каталог TP часто переименовывают, иногда это делается для расположения нескольких версий Паскаля на ЭВМ. Так, например, версия Турбо Паскаль 7.0 часто располагается в каталоге TP7. Начинать выполнение лабораторных работ следует с создания личного каталога, в котором будут храниться создаваемые программы до отчета преподавателю.

Редактирование программы пользователя происходит в **окне редактирования**, которое следует открыть. Для этого необходимо:

1. Нажать кнопку F10, при этом в строке меню (самая верхняя стока на экране со списком разрешенных опций) одна из опций выделится зеленым цветом, этим выделением можно управлять кнопками управления курсора.

2. Установить зеленый цвет на опцию **File**. Затем нажать клавишу Enter (Ввод) на клавиатуре. На экране появится список команд, определенных для этой опции.

3. Выбрать команду **New** в этом подменю и нажать клавишу Enter.

(Или вся эта последовательность команд естественным образом может выполниться мышью – самым распространенным персональным устройством манипуляторного типа).

На экране появится синее окно редактирования. В нем можно набирать и редактировать программу. Среда Турбо Паскаля является многооконной, то есть можно открывать окно редактирования для каждой новой программы и они будут содержаться в машинной памяти одновременно, не мешая друг другу.

Текст в окне можно смещать относительно листа с помощью клавиш:

PageUp – на страницу вверх;

PageDown – на страницу вниз;

Home – в начало текущей строки (домой);

End – в конец текущей строки;

Ctrl – PageUp – в начало текста;

Ctrl – PageDown – в конец текста.

Введенный символ можно стереть клавишей BackSpace (Забой). Клавиша Delete стирает символ, на который в данный момент указывает курсор. Комбинация клавиш Ctrl – Y стирает всю строку, на которой находится курсор. Чтобы «разрезать» строку, надо подвести курсор в место разреза и нажать клавишу Enter (Ввод). Чтобы «склеить» соседние строки, надо установить курсор в конец первой строки и нажать Delete или установить курсор в начало следующей строки и нажать клавишу BackSpace. Эти действия определяются тем, что конец строки кодируется на экране символом, обычно невидимым, то есть склейка – уничтожение этого невидимого символа, разрезание – его вставка.

По умолчанию в текстовом редакторе устанавливается режим вставки вводимых символов, в котором каждый вводимый символ «раздвигает» текст, смещая вправо остаток текста. В режиме замены символов новые символы накладываются на старые. Клавиша Insert – переход из режима вставки в режим замены, и наоборот. Следует

иметь в виду, что среда Турбо Паскаля запоминает установки последнего сеанса работы, поэтому, если Ваше рабочее место не является индивидуальным, нежелательно менять его, так как это может помешать работе лицам, использующим данное АРМ (автоматизированное рабочее место) для своей деятельности.

Признаком того, в каком режиме находится редактор, является внешний вид курсора. При разных установках это может быть: мигающий символ курсора, величиной в половину заполненного текстового символа – режим замены; крупный прямоугольник, заполняющий символ целиком – режим вставки, или наоборот.

Приведем несколько команд работы с блоками – частями текста программы без пропусков:

Ctrl – KB – пометка начала блока;

Ctrl – KK – пометка конца блока;

Ctrl – KH – удалить пометку блока;

Ctrl – KY – стирание блока;

Ctrl – KC – копирование блока;

Ctrl – KV – перемещение блока;

Ctrl – KW – запись блока в дисковый файл;

Ctrl – KR – чтение блока из дискового файла;

Ctrl – KP – печать блока;

(Команда типа Ctrl – KB выполняется так: при нажатой клавише Ctrl нажимается сначала клавиша K, затем клавиша K отпускается и нажимается клавиша B.)

Обычный язык состоит из слов, язык программирования – из команд (операторов). Как правило, основой оператора служит английское слово (или его сокращение), поясняющее смысл оператора. Кроме того, в Турбо Паскале большое значение имеют слова, значения которых определены самим программистом. Необходимость и правила введения таких слов описаны ниже.

Идентификаторы и служебные слова

Служебные (зарезервированные) слова – это ограниченная группа слов, построенных из букв. Каждое служебное слово представляет собой неделимое образование, смысл которого фиксирован в языке. Служебные слова НЕЛЬЗЯ использовать в качестве имен, вводимых программистом (то есть в качестве идентификаторов). В стандартной конфигурации среды программирования они выделяются белым цветом. Всего в Турбо Паскале 7.0 зарезервировано 55 служебных слов:

absolute	do	implementation	or	then
and	downto	in	packed	to
array	else	inline	private	type
asm	end	interface	procedure	unit
assembler	external	interrupt	program	until
begin	file	label	record	uses
case	for	mod	repeat	var
const	forward	nil	set	virtual
constructor	function	not	shl	while
destructor	goto	object	shr	with
div	if	of	string	xor

Идентификаторы в Турбо Паскале – это имена констант, переменных, меток, типов, объектов, процедур, функций, модулей, программ и полей в записях, то есть все те имена, которые программист может сам ввести или изменить их назначение. Они могут иметь произвольную длину, но значащими являются только первые 63 символа. Идентификатор всегда начинается буквой латинского алфавита или знаком подчеркивания и может содержать буквы латинского алфавита, цифры и знак подчеркивания. Пробел и специальные символы не могут входить в идентификатор.

Пример.

Идентификаторы	Не идентификаторы	Пояснения
a	1Program	Начинается с цифры
alpha	block#1	Содержит специальный символ
MyProgramBestProgram	My Program	Содержит пробел
date_27_sep_39	mod	Служебное слово
Exter		

_beta		

Правила набора и редактирования программы в Паскале

1. Строка в текстовом редакторе Турбо Паскаль может содержать произвольное количество операторов, но для удобства работы с программой лучше располагать их в столбик, по одному оператору в строке, используя отступы от края экрана для выделения отдельных блоков (подробнее об отступах будет рассказано при определении каждого отдельного оператора).

2. При наборе операторов используются только латинские буквы (не имеет значения заглавные или строчные, но желательно использовать строчные, выделяя заглавными буквами идентификаторы, на которые следует обратить внимание). Переключение с русского алфавита на латинский может осуществляться различными комбинациями клавиш, например: правая кнопка Shift, одновременное нажатие правой и левой кнопок Shift и т.п. Это определяется пользователем при настройке режима DOS .

3. Программа выполняется после нажатия комбинации клавиш Ctrl – F9 (то есть при нажатой кнопке Ctrl следует нажать кнопку F9).

4. Если хотя бы один символ в тексте программы набран неверно, на экран будет выдано сообщение об ошибке и курсор будет установлен на предполагаемое место ошибки.

5. После выполнения программы на экране снова появится текст файла с операторами. Для просмотра результатов вывода на экран следует нажать комбинацию клавиш Alt – F5.

6. Выход из среды: нажатие Alt – X. Перед нажатием следует сохранить свою работу в свой каталог.

Процедура вывода информации на экран

Рассмотрим некоторые операторы и процедуры Турбо Паскаля.

WRITELN (английское слово **Write** означает **писать**, **Ln** – сокращение от слова **Line** – **линия** (**строка**), суффикс **Ln** означает, что следующий вывод будет осуществляться в новую строку).

Процедура **writeln** – это команда компьютеру что-нибудь напечатать. Причем выведено может быть не только какое-либо сообщение или число, но и значение весьма сложного арифметического выражения. Например, можно напечатать значение суммы.

Для этого нужно:

1. Наберите следующий текст без столбца пояснений.

Программа	**ПОЯСНЕНИЯ**
BEGIN	**начало** – служебное слово, означающее начало исполняемой части программы
Writeln (123 + 456)**;**	тело программы
END.	**конец** – служебное слово, означающее конец исполняемой части программы. В конце программы ставится точка.

2. Нажмите комбинацию клавиш Ctrl – F9, чтобы программа выполнилась.

3. Нажмите комбинацию клавиш Alt – F5, чтобы увидеть результат: 579.

4. Вернитесь в программу нажатием на любую кнопку.

Правила записи арифметических выражений в программе

1. Все цифры находятся в верхнем ряду белых клавиш и на дополнительной клавиатуре справа. Для использования дополнительной клавиатуры при наборе чисел необходимо, чтобы был включен индикатор Num Lock. Если индикатор отключен, то нажмите кнопку Num Lock.

2. Перед отрицательными числами ставится минус.

3. В десятичных дробях вместо запятой ставится точка.

4. Для записи очень больших и малых по модулю чисел используется запись чисел с порядком. Так, число $1{,}02\square\square \cdot 10^{19}$ следует записать в виде 1.02e-19. Буква **e** заменяет число 10, знак умножения перед ней ставить не нужно.

Вывод действительных (вещественных) чисел на экран осуществляется также в виде числа с порядком, например, если результат выполнения операции есть 1,2, то на экране появится число

1.200000000E00 (число 0,0236 выведется в форме 2.36000000E-02). Целые числа выводятся в привычном виде.

Арифметические операции

\+ Сложение.

– Вычитание.

* Умножение.

/ Деление.

div Целочисленное деление.

mod Остаток от целочисленного деления.

Специальная операция для возведения в степень в Паскале отсутствует. Есть несколько способов вычислить x^y. Для натурального, не очень большого, y, можно перемножить x на себя y раз, то есть вычислить $\underbrace{x \cdot x \cdot x \cdot \ldots \cdot x}_{y \text{ раз}}$, для вещественного y воспользоваться формулой $x^y = e^{y \cdot \ln x}$.

Для записи арифметических выражений можно использовать круглые скобки. Действия в скобках выполняются в первую очередь. Порядок действий определяется обычными математическими правилами, то есть сначала выполняются умножение, деление и нахождение остатка от деления, а затем сложение и вычитание. Символы с клавиатуры на экран идут без подъемов и опусканий. Так же в линию записывают арифметические выражения. Например: $\frac{1}{3} \to 1/3$.

Часто используется форматный вывод информации на экран. Например, WriteLn(123 / 456 **:** 6 **:** 3), выведет на экран результат деления, выделив под все число 6 знаков, из которых 3 – под дробную часть (вывод на экран будет выглядеть: _ 0.270, где знак ‘_’ – пробел).

Если арифметическое выражение не помещается в строку на экране, то можно нажать клавишу Enter (ВВОД) и продолжить его набор в следующей строке. Знак переноса ставить не нужно.

Сообщения об ошибках

Транслятор Турбо Паскаля распознает ошибки, нарушающие внутреннюю логику языка. При отладке программы следует узнавать сообщения компилятора на самые распространенные ошибки начинающих программистов. Существует много различных сообщений об ошибках, причем эти сообщения, вообще говоря, зависят от контекста программы, то есть одно и то же сообщение может быть вызвано различными причинами. Наиболее часто встречающиеся ситуации приведены ниже. При получении одного из сообщений, исправьте ошибку и запустите программу снова. Общий список всех сообщений об ошибках смотрите в библиографическом списке [9].

Так, если в приведенной выше программе при наборе слова **BEGIN** допущена неточность, например, набрано **BEGIM**, то появится сообщение:

BEGIN expected (ожидается BEGIN).

Если при наборе слова writeln допущена неточность, например, набрано writel**h**, то появится сообщение

Unknown identifier (Неизвестный идентификатор).

Если случилось деление на ноль, то сообщения будут разными для целых и вещественных чисел.

Для целых – Devision by zero (Деление на ноль).

Для вещественных – Invalid floating point operation (Недопустимая операция с плавающей точкой).

Если результат слишком велик, то сообщения также будут разными для целых и вещественных чисел.

Для целых – Overflow in arithmetic operation (Переполнение в арифметическом действии).

Для вещественных – Constant out of range (Константа вне диапазона).

Если при наборе программы между операторами опущена точка с запятой, то появится сообщение:

«**;**» expected (ожидается точка с запятой).

Если в конце программы не поставлена точка, то появится сообщение:

Unexpected end of file (Неожиданный конец файла).

Приведенный список является неполным, но сообщения из него чаще других возникают у начинающих программистов. Полный список сообщений приведен в [9].

Стандартные арифметические функции

В Турбо Паскале предусмотрена возможность вычисления многих элементарных функций. Запись функций производится аналогично их записи в математике: имя функции и аргумент в скобках (обязательно). Стандартные функции (см. табл. 2) могут входить в арифметические выражения. В качестве аргумента также можно использовать арифметическое выражение.

Таблица 2

Стандартные арифметические функции Турбо Паскаля

Pascal	**Математическая функция**	**Пояснение**	
Abs(x)	$\|x\|$	Модуль числа x	
ArcTan(x)	arctg x	Арктангенс числа x, результат в радианах	
Cos(x)	cos x	Косинус числа x	аргумент задается
Sin(x)	sin x	Синус числа x	в радианах
Exp(x)	e^x	e в степени x	
Frac(x)	$\{x\}$	Дробная часть числа x	
Int(x)	$[x]$	Целая часть числа x, результат – вещественного типа.	
Ln(x)	ln x	Натуральный логарифм числа x	
Pi	3.1415926...	Число Пи	
Sqr(x)	x^2	x в квадрате	
Sqrt(x)	$\sqrt{x}$	корень квадратный из x	
Trunc(x)		Целая часть x	результат
Round(x)		Округление x до ближайшего целого	целого типа
Odd(x)	Определение нечетности целого числа	*True*, если x – нечетное, *False*, если x четное.	

Если необходимо вычислить функцию, которая не входит в набор стандартных, то нужно выразить ее с помощью знаков арифметических действий и стандартных функций.

Таблица 3

Формулы тригонометрических, логарифмических и показательных функций, не входящих в список стандартных и выраженных через них

Название	**Формула**
1	*2*
Тангенс *x*	$\operatorname{tg} x = \frac{\sin x}{\cos x};$
Котангенс *x*	$\operatorname{ctg} x = \frac{\cos x}{\sin x};$
Арксинус *x*	$\arcsin x = \operatorname{arctg} \frac{x}{\sqrt{1-x^2}};$
Арккосинус *x*	$\arccos x = \frac{\pi}{2} - \operatorname{arctg} \frac{x}{\sqrt{1-x^2}};$
Арккотангенс *x*	$\operatorname{arcctg} x = \frac{\pi}{2} - \operatorname{arctg} x;$
Логарифм числа *x* по основанию *a*	$\log_a x = \frac{\ln x}{\ln a};$
Возведение в степень *y* положительного числа *x*	$x^y = e^{y \cdot \ln x}.$

Комментарии

Один из методов упрощения работы с программами на Паскале заключается в документировании любых блоков программы и любых хоть сколько-нибудь самостоятельных фрагментов. Комментарии в программах могут располагаться в любом месте, однако они не могут разрывать на части служебные слова и идентификаторы. В комментариях указывается информация о назначении операторов и частей программы.

Комментарии в программах на Паскале представляют собой произвольный текст, ограниченный с двух сторон фигурными скобками – **{ }**, или вместо открывающей фигурной скобки **{** применяют пару символов **(***, а вместо закрывающей фигурной скобки **}** – пару символов ***)**.

Пример.

{ комментарий **}**

(* комментарий ***)**

Комментарии могут быть вложенными, то есть несколько комментариев можно объединить в один.

Пример.

```
(*   { комментарий № 1 }
     { комментарий № 2 }
     { комментарий № 3 }
     комментарий № 4 *)
```

или так

```
{    (* комментарий № 1 *)
     (* комментарий № 2 *)
     (* комментарий № 3 *)
     комментарий № 4 }
```

Комментарии очень удобно использовать для размещения в одном файле нескольких программ, относящихся к одной лабораторной работе, так как работа, как правило, сдается целиком, и отдельные программы, составляющие текст лабораторной, можно закомментировать, чтобы они не мешали выполнению следующих.

В тексте комментария могут присутствовать любые символы, но символ денежной единицы $, стоящий сразу же за открывающей скобкой, то есть пара {$, является зарезервированной комбинацией и означает, что далее следует директива для компилятора, поэтому такое сочетание символов следует избегать.

Модуль CRT
Процедуры и функции управления текстовым режимом экрана
Вывод текстовой информации на экран

Новые операторы позволят красиво размещать на экране самую различную информацию, но предварительно необходимо познакомиться с механизмом хранения в Турбо Паскале специальных процедур и функций. Для этого в Турбо Паскале используются модули. Модули – это библиотеки программных объектов языка Турбо Паскаль (констант, типов, переменных, подпрограмм), которые могут свободно использоваться другими программными объектами. Так, модуль Graph (с которым познакомимся позже) содержит процедуры и функции, позволяющие реализовать графические возможности компьютера. Подпрограммы модуля Printer обеспечивают вывод информации на любые виды принтеров, и так далее.

Процедуры и функции управления текстовым режимом экрана располагаются в модуле CRT. Из достаточно большого количества объектов этого модуля мы познакомимся только с некоторыми (**ClrScr –** очистка экрана, **gotoXY(*позиция, строка*)** – переход курсора в заданные позицию и строку, **readkey** – задержка выполнения программы до нажатия на любую клавишу, **delay(*время*)** – задержка выполнения программы на указанное в миллисекундах время).

Для работы с любыми модулями необходимо указать в разделе описания модулей их спецификацию:

USES U1, U2, U3;

где **USES** – служебное слово, **U1, U2, U3** – идентификаторы используемых модулей.

Раздел описания модулей располагается в начале программы (перед служебным словом **BEGIN**).

Пример.

```
USES CRT; {Спецификация модуля CRT}
BEGIN
  writeln(123 + 456);
  readkey {Функция из модуля CRT}
END.
```

Аргументом процедуры **WRITELN** кроме арифметического выражения может быть также текст, который обязательно заключается в апострофы. Печать текста начинается от позиции, в которой находится курсор.

CLRSCR – очистка экрана. Вся информация, которая была выведена до этой процедуры на экран, будет стерта. Следующий вывод осуществится в первую строку экрана, начиная с первой позиции.

Процедура **GOTOXY (*позиция, строка*)** устанавливает курсор в заданную параметрами **(*позиция, строка*)** позицию (столбец символов на экране) и строку. В стандартном режиме работы (активизированном при включении) на экране всего 25 строк по 80 символов в каждой строке. Основой оператора **GOTOXY** является английское словосочетание **GO TO**, которое означает «**перейти к**».

Функция **READKEY** осуществляет задержку выполнения программы до нажатия на любую клавишу.

Процедура **DELAY (*время*)** осуществляет задержку выполнения программы на указанное в миллисекундах время.

В Турбо Паскале можно располагать несколько операторов в одной строке. Они отделяются друг от друга точкой с запятой. После запуска программы операторы поочередно исполняются слева направо. То есть программы

USES CRT; BEGIN ClrScr; gotoXY(15, 10); writeln(‘Привет’); readkey **END.** или

```
USES
   CRT;
 BEGIN
  ClrScr; gotoXY (15, 10); Writeln (‘Привет’);
  READKEY
END.
```

будут работать совершенно точно одинаково.

Процедура **WRITELN** позволяет выводить на экран одновременно строковые и численные значения. Этим пользуются, например,

чтобы объяснить смысл выводимых данных. Так, в результате работы программы

```
USES CRT;
BEGIN
   ClrScr;
   Writeln (‘123 + 456 =’, 123 + 456)
END.
```

на экран выведется информация 123 + 456 = 579.

Значения позиции – X (столбца) и строки – Y в операторе **gotoXY** должны быть целыми и положительными. В противном случае появится сообщение об ошибке:

Type mismatch (Ошибочный тип).

Кроме того, если значение позиции больше 80 или значение строки больше 25, то сообщение об ошибке не выдается, но и оператор не выполняется. Тот же самый результат будет и в случае нулевых значений любого аргумента. Если выражение, взятое в качестве номера строки или столбца, окажется отрицательным, то на экране появится сообщение об ошибке:

Constant out of range (Константа вне диапазона).

В операторе **gotoXY** в качестве аргументов можно использовать арифметические выражения. В этом случае приходится контролировать ситуацию, в которой выражение принимает нецелое значение. Следует учитывать, что компьютер считает результат выполнения операции 4 / 2 нецелым (вещественным), так как операция ‘/’ – операция вещественного деления. Ее выполнение преобразует результат деления в вещественный тип, и нужно использовать для подобного рода вычислений специальные операции целочисленного деления (div, mod) или округлять результат функциями, возвращающими значения целого типа (Trunc(x) – целая часть x, Round(x) – округление x до ближайшего целого).

Одной процедурой **WRITELN** можно осуществлять вывод сразу нескольких результатов вычислений и печатей текстов. Для этого по-

сле слова **WRITELN** в скобках последовательно указываются арифметические выражения и тексты (каждый текст обязательно заключается в апострофы), отделенные один от другого запятой.

Печать безымянных результатов на экране может привести к путанице. Желательно печатать результаты с текстовой подсказкой.

После выполнения процедуры **WRITELN** осуществляется переход курсора в следующую строку. Перехода курсора в следующую строку не произойдет, если использовать процедуру **WRITE** (так называемая печать без перевода строки). Можно применять операторы **WRITELN** и **WRITE** без аргументов, то есть без выражений в скобках. Тогда скобки тоже пропускают. Процедура **WRITE** без аргументов оставляет экран вывода без изменений, а исполнение процедуры **WRITELN** без аргументов используют для пропуска строк на экране вывода.

Процедуры управления цветом символов в текстовых режимах

Монитор может работать в различных текстовых режимах. Рассмотрим только простейшие четыре. В этих режимах экран разделен на 25 строк, в каждой строке позиции для 40 или 80 символов. Процедуры управления тестовыми режимами находятся в модуле CRT.

Пример.

Наберите следующую программу:

```
USES CRT;
BEGIN
   TextMode(co40);
   Write (' Демонстрация ');
   Readkey;
   TextMode (co80);
   Write (' Демонстрация ');
   Readkey;
END.
```

Процедура **TextMode** выполняет переход из одного текстового режима в другой. В скобках указываются предопределенные имена констант (bw40, co40, bw80, co80) или значения этих констант (соответственно 0, 1, 2, 3).

Выполнение процедуры **TextMode** с параметром **bw40** приведет к установлению черно-белого режима с длиной строки 40 символов. Выполнение процедуры **TextMode** с параметром **co40** приведет к установлению цветного режима с длиной строки 40 символов. Выполнение процедуры **TextMode** с параметром **bw80** приведет к установлению черно-белого режима с длиной строки 80 символов. Выполнение процедуры **TextMode** с параметром **co80** приведет к установлению цветного режима с длиной строки 80 символов. Выполнение процедуры **TextMode** с параметром **LastMode** восстанавливает текстовый режим, который использовался последним.

В любом из текстовых режимов (как в цветных, так и в черно-белых) каждый символ может высвечиваться в одном из 16 цветов (значения от 0 до 15) на любом из 8 цветов (значения от 0 до 7) фона.

Определение цвета высвечиваемых символов осуществляется с помощью процедуры TextColor, а цвет фона – с помощью процедуры TextBackGround. Цвета можно устанавливать с помощью предопределенных констант или соответствующих им численных значений. В дальнейшем будем использовать только цветные текстовые режимы.

Таблица 4

Константы цвета

Значения предопределенной константы	**Предопределенная константа цвета**	**Цвета**
1	2	3
0	Black	черный
1	Blue	синий
2	Green	зеленый
3	Cyan	голубой

Окончание табл. 4

1	*2*	*3*
4	Red	красный

5	Magenta	фиолетовый
6	Brown	коричневый
7	LightGray	светло-серый
8	DarkGray	темно-серый
9	LightBlue	светло-синий
10	LightGreen	светло-зеленый
11	LightCyan	светло-голубой
12	LightRed	светло-красный
13	LightMagenta	светло-фиолетовый
14	Yellow	желтый
15	White	белый

Совершенно безразлично, что используется, константа или соответствующее ей значение. Например: константа **Green** или соответствующее ей значение **2**.

Пример.

Наберите и сравните работу двух представленных ниже программ:

1. **USES** CRT;
   BEGIN
      TextMode(co40);
      TextColor(Yellow);
      TextBackGround (Red);
      Writeln ('желтое, на красном фоне');
      Readkey;
   END.
2. **USES** CRT;
   BEGIN
      TextMode(1);
      TextColor(14);
      TextBackGround (4);
      Writeln ('желтое, на красном фоне');
      Readkey;
   END.

Если к числу, определяющему цвет, добавить 16, то высвеченный символ будет мигать. В программе это можно выразить, используя предопределенное имя константы Blink со значением 16.

Завершено предварительное знакомство с компьютером. Вы освоились с клавиатурой, научились вычислять значения арифметических выражений, познакомились с простейшими процедурами и функциями Турбо Паскаля. Эти знания послужат основой для дальнейшего изучения языка программирования и компьютера.

Порядок выполнения работы

1. Узнайте номер и цветовую гамму своего варианта.
2. Создайте документ «Титульный лист».
3. Запустите программу Турбо Паскаль.
4. Подготовьте текст задания, подставив значение варианта в свои примеры.
5. Выполните задание в цветовой гамме, указанной преподавателем.
6. Отладьте программу (исключите все сообщения об ошибках).
7. В текстовом редакторе WORD или рукописно создайте отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.
2. Задания варианта.
3. Описание использованных в работе операций.
4. Программные единицы.
5. Результат выполнения программы.
6. Описание использовавшихся операций копирования строк.

Варианты задач

Учащийся получает номер варианта с цветовой гаммой. Цветовая гамма может быть определена, например, по номеру его в списке группы N, если вариант номер 15, можно использовать число 8: цвет букв N mod 8, цвет фона N mod 16. При совпадении номеров цвета фона и символов, осуществить сдвиг на единицу одного из цветов.

Вообще номера вариантов заданий могут быть распределены произвольно по списку. Конкретные числа в лабораторной 1 следует вычислять в зависимости от номера варианта. Так, пусть, например, студент получает вариант с номером *n*. Значения в примерах своего варианта можно вычислить, а можно ввести выражение, использующее переменную *n*.

Запись 0, *nn* означает дробное число – ноль целых, *nn* десятых (то есть для второго варианта – 0,0202, для двенадцатого варианта – 0,1212).

Если арифметическое выражение не помещается в строку на экране, то можно нажать клавишу Enter (ВВОД) и продолжить его набор в следующей строке. Знак переноса ставить не нужно.

1)	$\frac{1}{n}+\frac{1}{n+1}+\frac{1}{n+2}+\frac{1}{n+3}+\frac{1}{n+4}$;
2)	$n^{n-15} - (\lvert n - 15\rvert)^n$;
3)	$n^2 + (n + 1)^2 + (n + 2)^2 + (n + 3)^2 + (n + 4)^2$;
4)	$\frac{n^3+(n+1)^3+(n+2)^3}{(n+5)^3-(n+3)\cdot(n+4)\cdot(n+5)}$;
5)	$\frac{n+(n+1)}{(n+2)+(n+3)} : \frac{(n+4)+(n+5)}{(n+6)+(n+7)}$;
6)	$n^{\frac{2}{3}} - n^{\frac{3}{2}}$;
7)	$\sqrt{(n+8)^2+(n+4)^2}$;
8)	$\sqrt{\sqrt[3]{n}+\sqrt[4]{n}}$;
9)	$625^{-0,nn}$;
10)	$\frac{\sqrt[5]{(n+6)}}{\left(1+\sqrt{n+5}\right)^4}$;
11)	$\log_n(n+1)$;
12)	tg(*n* – 5), аргумент функции в радианах;
13)	cos *x* при $x = n^o$, аргумент функции в градусах;
14)	$e^{-\frac{1}{n}}$;
15)	arccos 0,*nn*, результат должен быть в градусах

Решение типового варианта

Рассмотрим пример оформления работы.

Пусть вариант студента № 31, цвет фона = 31 mod 8 = 7, цвет букв = 31 mod 16 = 15.

Сформируем отчет к лабораторной работе № 1.

1. Титульный лист, сформируем обязательные компоненты для любой лабораторной работы, любого варианта.

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Орский гуманитарно-технологический институт (филиал)
Государственного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»
КАФЕДРА: МАИТ и МОИ.
КОД СПЕЦИАЛЬНОСТИ 032200
СЕМЕСТР 1
Лабораторная работа 1
ПО ДИСЦИПЛИНЕ
«Информатика»
ВАРИАНТ 31
Выполнил студент группы 1А Петрова Л. К.
Проверил ассистент каф. МАИТ и МОИ Сидоров Н. Е.
Дата выполнения работы 12/09/07

2. Сформулируем задание варианта 31.

1)	$\frac{1}{31}+\frac{1}{32}+\frac{1}{33}+\frac{1}{34}+\frac{1}{35}$;
2)	$31^{16}-16^{31}$;
3)	$31^2+32^2+33^2+34^2+35^2$;
4)	$\frac{31^3+32^3+33^3}{36^3-34\cdot35\cdot36}$;
5)	$\frac{31+32}{33+34}:\frac{35+36}{37+38}$;
6)	$31^{\frac{2}{3}}-31^{\frac{3}{2}}$;
7)	$\sqrt{39^2+35^2}$;
8)	$\sqrt{\sqrt[3]{31}+\sqrt[4]{31}}$;
9)	$625^{-0,3131}$;
10)	$\frac{\sqrt[5]{37}}{\left(1+\sqrt{36}\right)^4}$;
11)	$\log_{31}32$;

12)	tg 26 радиан;
13)	cos 31^o;
14)	$e^{-\frac{1}{31}}$;
15)	arccos 0,3131

3. Опишем использованные в работе операции.

При выполнении лабораторной работы были применены следующие формулы и стандартные арифметические функции:

а) $x^y = e^{y \cdot \ln x}$ б) sqr(x) в) $x^y = e^{y \cdot \ln x}$

г) sqrt(x) д) $x^y = e^{y \cdot \ln x}$ е) $\log_a x = \frac{\ln x}{\ln a}$

ж) $\operatorname{tg} x = \frac{\sin x}{\cos x}$ з) x радиан $\to x \cdot \frac{180}{\pi}$ градусов

и) $\arccos x = \frac{\pi}{2} - \operatorname{arctg} \frac{x}{\sqrt{1-x^2}}$

4. Все 12 программных единиц реализованы в одной программе.

```
Begin
  TextBackGround (2);
  TextColor (0);
  Writeln('1)',1/31 + 1/32 + 1/33 + 1/34 + 1/35);
  Writeln('2)',exp(16*ln(31))-exp(31*ln(16)));
  Writeln('3)',sqr(31)+sqr(32)+sqr(33)+sqr(34)+sqr
(35));
  Writeln('4)',(31*31*31+32*32*32+33*33*33)/(36*36
*36-34*35*36));
  Writeln('5)',(31+32)/(33+34)/((35+36)/(37+38)));
  Writeln('6)',exp(2/3*ln(31))-exp(3/2*ln(31)));
  Writeln('7)',sqrt(sqr(39)+sqr(35)));
  Writeln('8)',exp(1/3*ln(31))-exp(1/4*ln(31)));
  Writeln('9)',exp(-0.3131*ln(625)));
  Writeln('10)',exp(1/5*ln(37))/(1+sqrt(36))/(1+
sqrt(36))/(1+sqrt(36))/(1+sqrt(36)));
  Writeln('11)',ln(32)/ln(31));
```

```
  Writeln('12)',sin(26)/cos(26));
  Writeln('13)',cos(31*180/pi));
  Writeln('14)',exp(-1/31));
  Writeln('15)',pi/2-arctan(0.3131/sqrt(1-sqr(0.
3131))));
end.
```

5. Результат выполнения программы.

1) 1.5179428810E-01

2) -2.1267647934E+37

3) 5455

4) 2.5811320755E+01

5) 9.9327307126E-01

6) -1.6273242285E+02

7) 5.2402290026E+01

8) 7.8176959063E-01

9) 1.3323243540E-01

10) 8.5752775364E-04

11) 1.0092454329E+00

12) 1.1787535543E+00

13) -3.9078173083E-01

14) 9.6825667714E-01

15) 1.2523409236E+00

6. Описание использовавшихся операций копирования строк.

Введем программу, исполняющую задание 1 в указанной цветовой гамме.

```
Begin
  TextBackGround (7);
  TextColor (15);
  Writeln (' 1) ',1/31 + 1/32 + 1/33 + 1/34 + 1/35);
End.
```

Затем выделим строку с оператором Writeln, для этого установим курсор на начало строки, нажмем клавишу Shift и, не отпуская ее, выделим нужный фрагмент клавишами управления курсора. Про-

ще всего выделить всю строку, просто нажав клавишу <вниз> [↓]. Затем переставим курсор в позицию на экране, где должна появиться копия выделенного и нажмем Ctrl – KC. Программа приобретет вид:

```
Begin
  TextBackGround (2);
  TextColor (0);
  Writeln (' 1) ',1/31 + 1/32 + 1/33 + 1/34 + 1/35);
  Writeln (' 1) ', 1/31 + 1/32 + 1/33 + 1/34 + 1/35);
End.
```

Затем исправим строку со вторым оператором writeln так, чтобы она приобрела вид:

```
Writeln (' 2) ', exp (16 * ln (31))-exp (31 * ln (16)));
```

Аналогично выполним оставшиеся задания.

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. Книга 1. Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Абрамов, С. А. Начала информатики / С. А. Абрамов, Е. В. Зима. – М. : Наука, 1989. – 256 с.

3. Турбо Паскаль 7.0 – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

4. Рубенкинг, Н. Турбо Паскаль для Windows : в 2 т. / Н. Рубенкинг ; пер. с англ. – Т. 1. – М. : Мир, 1993. – 536 с., ил.

Вопросы для самопроверки

1. Назовите имя создателя языка программирования Паскаль, год и страну создания.

2. Назовите имя создателя языка программирования Турбо Паскаль 1.0 и первоначальное название этой реализации языка Паскаль.

3. Сформулируйте связь языков программирования PASCAL и Турбо Паскаль.

4. Укажите общее назначение использования функциональных клавиш F1 – F12 в языке Турбо Паскаль.

5. Опишите последовательность действий для получения справки по конкретному оператору или константе Турбо Паскаля.

6. Опишите последовательность действий для сохранения файла программы Турбо Паскаля.

7. Перечислите клавиши, обеспечивающие пошаговое прохождение программы, укажите действие каждой такой клавиши.

8. Сформулируйте отличие в действиях компилятора при нажатии клавиш Alt – F9 и клавиш Ctrl – F9.

9. Укажите действия (любую последовательность), приводящие к выходу из среды Турбо Паскаля.

10. Опишите последовательность действий (любую) для входа в интегрированную среду Турбо Паскаля.

11. Опишите последовательность действий (любую) позволяющую копировать фрагмент внутри программы в интегрированной среде Турбо Паскаль.

12. Дайте определение идентификатора. Приведите три примера идентификаторов.

13. Дайте определение служебного слова. Приведите три примера служебных слов.

14. Укажите максимальное количество операторов, допустимое для размещения в одной строке.

15. Опишите способы, позволяющие осуществить просмотр результатов вывода на экран в интегрированной среде Турбо Паскаль.

16. Дайте описание формата использования процедуры **WRITE** (вывод информации на экран). Приведите пример использования процедуры **WRITE.**

17. Дайте описание формата использования процедуры **WRITELN** (вывод информации на экран). Приведите пример использования процедуры **WRITELN.**

18. Дайте описание использования форматного вывода информации на экран. Приведите пример форматного вывода на экран значения вещественной переменной с тремя знаками после запятой.

19. Перечислите некоторые стандартные арифметические функции в Паскале (не менее пяти). Напишите выражение на Паскале с использованием стандартных арифметических функций, вычисляющее *TG* 5 + *LN* 3.

20. Укажите назначение комментариев в программе на языке Паскаль. Дайте условные обозначения комментариев в программе на языке Паскаль.

21. Перечислите арифметические операции на языке Паскаль. Проиллюстрируйте каждую операцию на конкретных значениях.

22. Дайте описание формата использования процедуры **READ** (ввод данных). Приведите пример использования процедуры **READ.**

23. Дайте описание формата использования процедуры **READLN** (ввод данных). Приведите пример использования процедуры **READLN.**

24. Назовите любые две подпрограммы, размещенные в модуле CRT. Сформулируйте общую характеристику всех подпрограмм модуля CRT.

25. Укажите процедуру, осуществляющую очистку экрана в текстовом режиме.

26. Сформулируйте отличие в использовании предопределенной константы цвета от использования соответствующего ей значения.

ЛАБОРАТОРНАЯ РАБОТА 2.
СТРУКТУРА ТИПОВ ДАННЫХ И ВЫЧИСЛЕНИЕ ВЫРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПЕРЕМЕННЫХ (2 ЧАСА)

Цель работы: иучение правил построения и вычисления значения выражения с использованием переменных, стандартных функций и арифметических операций.

Теоретические положения

Определение переменных в программе

В дальнейшем идентификаторы, которые могут менять свои значения по ходу выполнения программы, будем называть переменными. ПЕРЕМЕННАЯ – отмеченная именем область в памяти компьютера, в которую могут записываться различные значения. В памяти компьютера переменная создается в результате объявления (описания) ее в специальных разделах, которые начинаются служебным словом **VAR**. При описании следует указывать имя переменной и тип значений, которые может принимать переменная. Из всего многообразия типов, определенных в Турбо Паскале 7.0, для выполнения лабораторных работ достаточно использовать **INTEGER** (целый), **REAL** (действительный), **CHAR** (символьный), **STRING** (строковый). **BOOLEAN** (логический).

Таблица 1

Типы переменных	**Диапазоны значений**
INTEGER	(целый) позволяет переменной принимать целочисленные значения в диапазоне от –32768 до 32767.
REAL	(действительный) позволяет переменной принимать вещественные значения. Переменные типа REAL содержат действительные значения, мантисса которых 11-12 цифр, а диапазон десятичного порядка от –39 до 38.
CHAR	(символьный) переменная этого типа содержит один символ.
STRING	(строковый) тип данных, отсутствовавший в стандартном Паскале, принимает значения строки символов, при описании переменной этого типа можно указать в квадратных скобках максимальное количество символов. Если этот параметр отсутствует, то максимальное количество символов в строке равно 255.
BOOLEAN	(логический) переменная этого типа принимает значения FALSE (ложь) или TRUE (истина)

Общее представление о разветвленной структуре типов данных Турбо Паскаля, подробное описание которой занимает большой объем, можно получить, изучив следующий рисунок:

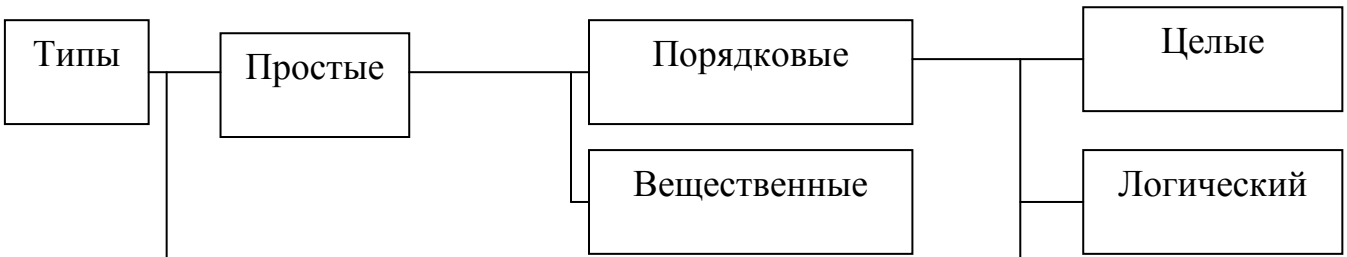


Рис. 1. Структура типов данных Турбо Паскаля

В программе описание выглядит следующим образом: раздел описаний начинается служебным словом VAR, затем через пробел или в другой строке – имя переменной (или несколько переменных, разделенных запятой), а через двоеточие – тип переменной (переменных).

Пример.

```
Var
   sigma: real;
   a,b,c,d: char;
   text1: string[15];
   text2: string;
```

Раздел описания переменных должен располагаться до тела основной программы. Всего в программе Турбо Паскаля могут быть определены следующие разделы:

– заголовок программы (**program**);

– раздел объявления используемых модулей (**uses**);

– раздел объявления меток (**label**);

– раздел объявления констант (**const**);

– раздел объявления типов (**type**);

– раздел объявления переменных (**var**);

– раздел объявления процедур и функций (**procedure**, **function**);

– тело программы (обязательная часть).

Любой из этих заголовков, кроме тела программы, может отсутствовать.

Оператор присваивания

Уже известно, что компьютер легко вычисляет значения арифметических выражений. Очень удобные для вычислений процедуры WRITE и WRITELN обладают существенным недостатком. Они печатают результат на экран, но не могут сохранить его в памяти компьютера. Этим результатом нельзя воспользоваться для дальнейших вычислений внутри программы. Для вычислений с сохранением результатов в памяти компьютера используется оператор присваивания. Он имеет вид:

***x*: = *A*,**

где ***x*** – имя переменной, в которой будет сохранен результат (слева от знака **:=**);

:= – Знак присваивания, читается: «присвоить»;

A – Арифметическое выражение, значение которого вычисляется.

Пример.

```
x := sin (b + c) / 2;
c := a * b;
A := 2.5 + A;
```

Процедуры ввода информации

Очень часто бывает, что программу пишут для работы со значениями, которые заранее неизвестны. Программа должна их запросить у человека непосредственно в процессе своей работы.

READ, **READLN** – процедуры ввода считывают данные с клавиатуры. Разница между этими процедурами в том, что **READLN** за-

крывает строку ввода, и следующее считывание, или вывод на экран, произойдет в следующей строке.

Представление алгоритмов

Понятие алгоритма – одно из фундаментальных понятий информатики. Алгоритмизация наряду с моделированием выступает в качестве общего метода информатики. К реализации определенных алгоритмов сводятся процессы управления в различных системах, что делает понятие алгоритма близким и кибернетике.

Алгоритмы являются объектом систематического исследования пограничной между математикой и информатикой научной дисциплины, примыкающей к математической логике – теории алгоритмов.

Особенность положения состоит в том, что при решении практических задач, предполагающих разработку алгоритмов для реализации на ЭВМ, и тем более при использовании на практике информационных технологий, можно, как правило, не опираться на высокую формализацию данного понятия. Поэтому представляется целесообразным познакомиться с алгоритмами и алгоритмизацией на основе содержательного толкования понятия алгоритма и рассмотрения основных его свойств. При таком подходе алгоритмизация более выступает как набор определенных практических приемов, особых специфических навыков рационального мышления в рамках заданных языковых средств.

Промышленное программирование и изучение алгоритмизации в школе ставит перед собой во многом абсолютно разные задачи. Промышленное программирование требует однозначности восприятия алгоритма, по истечении любого количества лет. Отсюда жесткая регламентация при создании документов по стандартам алгоритма соответствующего года создания. В качестве примера можно ознакомиться с Государственными стандартами. Легко увидеть, что разработанные стандарты очень искусственно позволяют отразить особенности конкретных языков программирования. Именно поэтому специалисты по обучению конкретным языкам программирования высо-

кого уровня не перестают искать новые способы визуализации алгоритмов.

Для школ разработаны способы графического представления алгоритмов, так называемые блок-схемы. Этот способ имеет ряд преимуществ, благодаря наглядности, обеспечивающей, в частности, высокую «читаемость» алгоритма и явное отображение управления в нем. Мы будем использовать именно школьный набор схем управления, так как студент легко может независимо от конкретной лабораторной работы обновить свои познания по этому вопросу.

Структурное программирование

Практика программирования показала необходимость научно обоснованной методологии разработки и документирования алгоритмов и программ. Эта методология должна касаться анализа исходной задачи, разделения ее на достаточно самостоятельные части и программирования этих частей по возможности независимо друг от друга. Такой методологией является структурное программирование. По своей сути оно воплощает принципы системного подхода в процессе создания и эксплуатации программного обеспечения ЭВМ. В основу структурного программирования положены следующие достаточно простые положения:

1. Алгоритм и программа должны составляться поэтапно (по шагам).

2. Сложная задача должна разбиваться на достаточно простые, легко воспринимаемые части, каждая из которых имеет только один вход и один выход.

3. Логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур. Использование этих положений позволяет внести определенную систему в труд программиста и составлять удобочитаемые алгоритмы (и программы), которые легко изучать и проверять. Фундаментом структурного программирования является теорема о структурировании. Эта теорема устанавливает, что, как бы сложна ни была задача,

схема соответствующей программы всегда может быть представлена с использованием весьма ограниченного числа элементарных управляющих структур. Элементарные структуры могут соединяться между собой, образуя более сложные структуры, по тем же самым элементарным схемам. Базовыми элементарными структурами являются структуры, представленные на рисунке 2.

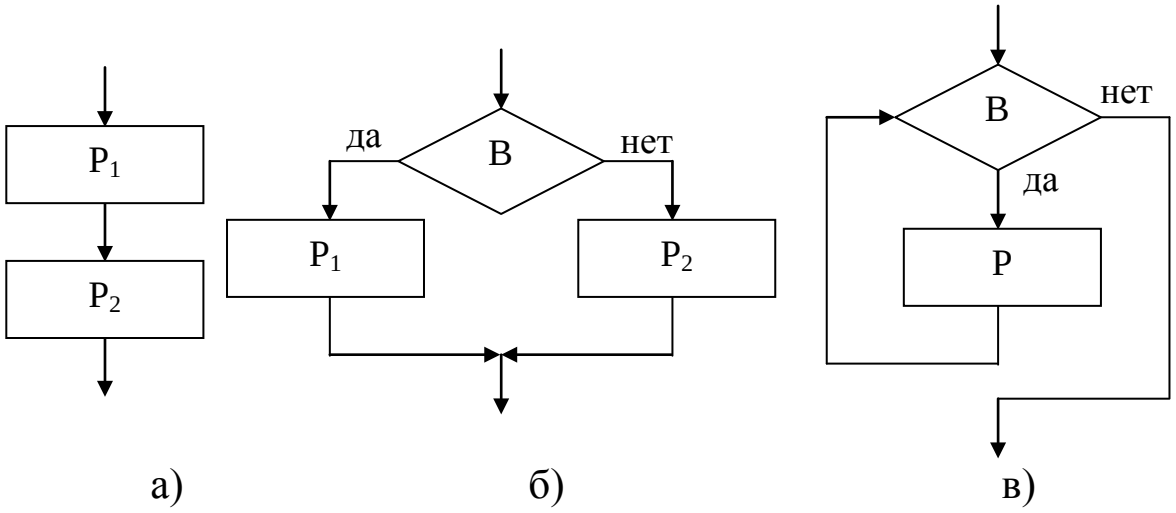


Рис. 2. Базовые элементарные структуры

Они обладают функциональной полнотой, то есть любой алгоритм может быть реализован в виде композиции этих трех конструкций. Каждая из конструкций имеет свое название. Так, первая их них (рис. 2, а) называется структурой типа последовательность (или просто последовательностью), вторая (рис. 2, б) – структурой выбора (разветвлением), третья (рис. 2, в) – структурой цикла с предусловием. При словесной записи алгоритма указанные структуры имеют соответственно следующий смысл: «выполнить P_1; выполнить P_2», «если B, то выполнить P_1, иначе выполнить P_2», «до тех пор, пока B, выполнять P», где B – условие; P, P_1, P_2 – действия.

Рассматривая схему программы, можно выделить в ней части (фрагменты), достаточно простые и понятные по структуре. Представление этих фрагментов укрупненными блоками существенно облегчает восприятие алгоритма (а в дальнейшем и программы) в целом.

Достаточно часто структурное программирование подразумевает использование более трех базисных структур. Применительно к

языку Паскаль, в котором наиболее полно нашли свое отражение идеи структурного программирования, целесообразно при проектировании алгоритмов дополнительно использовать еще четыре элементарные структуры (рис. 3):

– сокращенную запись разветвления (рис. 3, а);

– структуру варианта (рис. 3, б);

– структуру повторения или цикла с параметром (рис. 3, в);

– структуру цикла с постусловием (рис. 3, г).

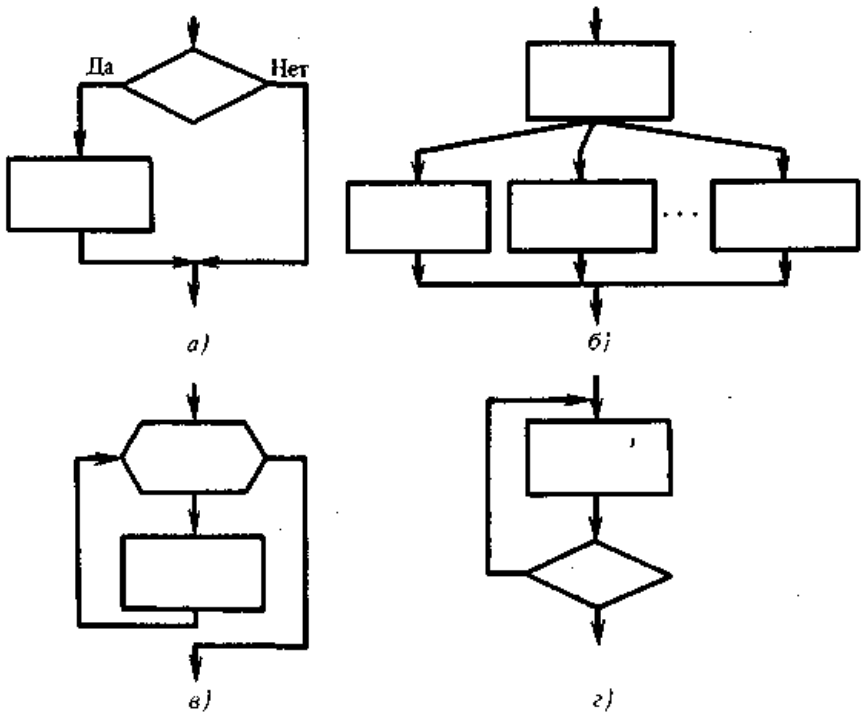


Рис. 3. Дополнительные элементарные структуры

Каждая из структур, показанных на рисунках, имеет один вход и один выход. В языке Паскаль имеются средства (операторы), позволяющие непосредственно реализовать в программе любую из этих структур, поэтому правильное использование типовых структур в процессе разработки алгоритма обеспечивает упрощение последующих этапов решения задачи на ЭВМ. В лабораторных работах № 2, 3 исследуется структура последовательность, № 4 исследуется структура цикл, № 5 исследуется структура ветвление. Здесь следует уточнить, что российская методика преподавания программирования зарождалась в условиях недостатка литературы по предмету, а значит,

инициативные россияне самостоятельно изыскивали иностранные издания, на уровне своих представлений о предмете переводили их и адаптировали перевод к действительности. Часто это были узкие специалисты, и необязательно программирования или преподавания. К тому же эти переводы изначально часто не предназначались для общероссийского пользования. То есть каждый специалист сам для себя формировал среду идеального изучения предмета. Отсюда большое количество терминов, которые означают одно и то же. Так, последовательность, следование или композиция – термины, которые прижились для обозначения линейной структуры. Лишь с приходом в школьную программу информатики начала складываться единая терминология, но до сих пор полезно бывает знать все устоявшиеся в практике названия структур.

Структурограммы

Способ изображения алгоритма с помощью структурограммы (схемы Насси – Шнейдермана) реализует в себе требования структурного программирования в схемах алгоритмов. Он позволяет изображать схему передач управления не с помощью явного указания линий потоков информации, а с помощью представления вложенности структур. Некоторые из используемых в этом способе графических символов (блоков) соответствуют изображению символов на схемах. Для изображения алгоритмов допускается использование следующих блоков:

1. Блок обработки (вычислений). Каждый символ структурограммы является блоком обработки. Каждый прямоугольник внутри любого символа представляет собой также блок обработки.

2. Блок следования. Этот символ объединяет ряд следующих друг за другом блоков обработки.

3. Блок решения. Этот символ применяется для обозначения структуры типа разветвления. Условие располагается в верхнем треугольнике, варианты решения – по сторонам треугольника, а процессы обработки обозначаются прямоугольниками.

4. Блок варианта. Этот символ представляет собой расширение блока решения. Те варианты выхода из этого блока, которые можно сформулировать точно, размещаются слева. Остальные объединяются в один, называемый выходом по несоблюдению условий, и располагаются справа. Если можно перечислить все возможные случаи, правую часть можно оставить незаполненной или совсем опустить.

5. Блок цикла с предусловием. Этот символ обозначает циклическую конструкцию с проверкой условия в начале цикла. Условие продолжения цикла размещается в верхней полосе, сливающейся с левой полосой, указывающей границу цикла. Данная структура может быть использована также для обозначения цикла с параметром. При этом вверху указывают закон изменения параметра цикла.

6. Блок цикла с постусловием. Этот символ аналогичен блоку цикла с предусловием, но условие располагается внизу. Это условие окончания цикла.

Каждый блок имеет форму прямоугольника и может быть вписан в любой внутренний прямоугольник любого другого блока. Блоки заполняются элементами словесной записи тем же способом, что и алгоритмы, изображенные с помощью схемы, то есть с помощью предложений на естественном языке или с использованием математических обозначений.

В ряде случаев для упрощения и наглядности алгоритма применяются модификации структурограмм. Для этого используют символы структурограмм одновременно с символами схем или символами других способов описания алгоритмов. В структурограмме могут быть отдельно изображены блоки ввода исходных данных и печати результатов, а также упрощено изображение блока решения с условием.

Примеры применения структурограмм рассматриваться не будут, так как их применение выходит за рамки данного практикума и школьной программы, хотя их внешний вид для основных структур будет указываться параллельно с блок-схемой.

Псевдокод

Псевдокод представляет собой метод применения естественного языка для описания алгоритма, но с конструкциями, близкими к конструкциям структурных языков программирования. Это позволяет создавать машинно-независимое описание процесса решения задачи, допускающее последующее использование языков программирования. Псевдокод состоит из такщгщ описания действий, которые могла бы выполнить ЭВМ, но которые содержат фразы в свободной форме. Основные команды псевдокода аналогичны командам для ЭВМ, то есть операторам языка программирования. Они комбинируются стандартным способом, в результате чего образуются более сложные команды и предложения. Псевдокод отличается от обычных детализированных устных алгоритмов (словесной формы записи) стандартизацией конструкций, более четкими описаниями, использованием ключевых (служебных) слов, строгим оформлением. Ключевые слова выбираются так, чтобы сделать алгоритм ясным, строгим и однозначным. Ключевые слова псевдокода обязательно выделяются подчеркиванием, более крупным шрифтом (написанием), использованием иностранных слов и тому подобное.

Примером псевдокода является алгоритмический язык академика А. П. Ершова, который изучается в старших классах средней школы. Ключевые слова алгоритмического языка выделяются более жирным написанием и подчеркиванием. В качестве основных в нем используются конструкция последовательности, конструкция если – то – иначе, конструкция повторения пока – нц – кц (служебные слова нц и кц служат для выделения в алгоритме тела цикла). Примеры записи алгоритмов на языке А. П. Ершова широко используются для обучения в школе, если его конструкции легче для восприятия, то допустимо программировать на нем, затем формально перевести на Паскаль. Соответствие конструкций алгоритмического языка и Паскаля легко выстроить самому.

Очень серьезные возражения против использования блок-схем можно сформулировать следующим образом: располагая операторы

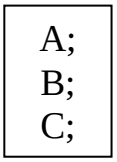
так называемой лесенкой, можно отразить алгоритм с той же или даже большей степенью детализации. Это во-первых.

Во-вторых, структурированные алгоритмы и программы, как правило, намного проще читать, понимать и исправлять, чем блок-схемы.

В-третьих, для структурированных алгоритмов и программ имеются систематические методы анализа правильности и выявления ошибок.

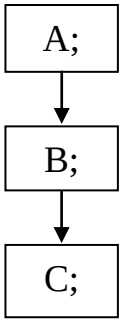
Единственная причина, по которой представление о графическом представлении алгоритма следует признать обязательным, – это то, что психолого-педагогические исследования показали, что для большого процента человечества рисованный объект всегда понятнее, чем его символьное описание. Начинающий программист опытным путем должен для себя определить, как ему проще создать алгоритм решения задачи: сначала блок-схему и по ней программу или сначала программу и по ней блок-схему. Второй способ категорически не рекомендуется теоретиками информатизации общества, но для практиков чаще бывает, что он быстрее, а значит эффективнее. Задачи, представленные в лабораторных работах № 1 и № 2, в основе своей имеют единственную структуру: последовательность.

На языке блок-схем и в структурных схемах по Насси и Шнейдерману структура последовательности может быть изображена как одинаково, так и по-разному. То есть, если последовательность команд изображается в одном блоке, это делается и там, и там так:

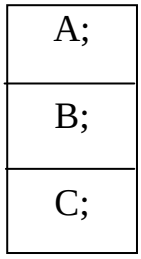


Если каждая команда размещается в отдельном блоке, это выглядит так.

На языке блок-схем:



В структурограммах по Насси и Шнейдерману:



В блок-схемах управление передается в направлении стрелки. Если стрелки отсутствуют, то передача осуществляется сверху вниз, или слева направо. Стандартное оформление структур (цикл, развилка – об этих структурах речь пойдет ниже) тоже позволяет не указывать направление перехода от блока к блоку.

В структурограммах по Насси и Шнейдерману управление всегда сверху вниз от структуры к структуре.

Порядок выполнения работы

1. Уточните номер своего варианта.

2. Ознакомьтесь со стандартами оформления блок-схем (см. теоретические положения лабораторной работы 2).

3. Составьте блок-схему алгоритмов заданий варианта.

4. Напишите программу по составленной блок-схеме.

5. Отладьте программу (исключите все сообщения об ошибках) и подберите значения для тестирования программ (запишите результаты тестирования).

6. В текстовом редакторе WORD или рукописно оформите отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.

2. Задания варианта

3. Составить блок-схемы алгоритмов заданий варианта.

4. Программные единицы.

5. Результат выполнения программы.

6. Описание использовавшихся операций копирования строк.

Варианты задач

	Вариант 1
1.	Напишите программу для вычисления объема цилиндра, если заданы значения

	радиуса основания цилиндра и высота, по формуле: $V_ц = \pi \cdot R^2 h$.
2.	Даны вещественные числа *a*, *b*, *c*, *d*, *x*. Составьте программу вычисления значения *y* по формуле $y = \frac{ax-b}{cx+d} + \cos\frac{ax-b}{cx+d}$.
	Вариант 2
1.	Напишите программу для вычисления объема шара, если задан радиус, по формуле: $V_ш = \frac{4\pi \cdot R^3}{3}$.
2.	Составьте программу вычисления значения *y* по формуле $y = 6x^4 + 5x^2 + 6x + 4 - \sin(\operatorname{tg} x^3)$.
	Вариант 3
1.	Напишите программу для вычисления объема конуса, если заданы радиус основания конуса и высота, по формуле: $V_к = \frac{\pi \cdot R^2 h}{3}$.
2.	Даны вещественные числа *a*, *c*, *x*. Составьте программу вычисления значения *y* по формуле $y = \frac{4{,}5cx-1{,}5a}{3c+a} - \operatorname{ctg}\frac{3c+a}{4{,}5cx-1{,}5a}$.
	Вариант 4
1.	Дано вещественное число *x*. Напишите программу, которая вычисляет значение *y* по формуле $y = \sin^3 x + \arccos\frac{x^2}{x^2+1}$.

2.	Даны вещественные числа *a*, *x*. Составьте программу вычисления значения *y* по формуле $y = \frac{1-\frac{2a}{1+x}}{1+\frac{2a}{1+x}}$.
	Вариант 5
1.	Дано вещественное число *x*. Напишите программу, которая вычисляет значение функции $y = \operatorname{tg} x^2 + \operatorname{arcctg}^2 \frac{x}{2}$.
2.	Дано вещественное число *x*. Составьте программу вычисления значения *y* по формуле $y = \frac{(x-2)(x^2-2x-15)}{(x+10)(x^2-x-20)} + \frac{x+3}{x+4}$.
	Вариант 6
1.	Даны действительные числа *a* и *b*. Вычислить $\sqrt[3]{ab\sin b}$.
2.	Дано вещественное число *x*. Составьте программу вычисления значения *y* по формуле $y = (x+1)^2 + 3(x+1) + \arccos\frac{x}{x+1}$.
	Вариант 7
1.	Даны действительные числа *x* и *y*. Получить $\left(\frac{\lvert x\rvert - \lvert y\rvert}{1+\lvert xy\rvert}\right)^{\frac{2}{xy}}$.
2.	Дано вещественное число *x*. Составьте программу вычисления значения *y* по

	формуле $y = 6x^2 + 3(x^2 + 1) - \arcsin\dfrac{x^2}{x^2+1}$.
	Вариант 8
1.	Дана длина ребра куба. Найдите объем куба и площадь его боковой поверхности. Использовать формулы: $V_к = a^3$, $S_{бп} = 4a^2$.
2.	Дано вещественное число x. Составьте программу вычисления значения y по формуле $y = 2(x + 3) + 3(x + 3)^2 + 5(x + 3)^3$.
	Вариант 9
1.	Даны три действительных положительных числа. Найдите среднее арифметическое квадратов этих чисел. Формула среднего арифметического = $= \dfrac{\text{сумма чисел}}{\text{количество чисел}}$.
2.	Дано вещественное число x. Составьте программу вычисления значения y по формуле $y = x^2\left(x^2 + 1\right) + \sqrt{x}$.
	Вариант 10
1.	Даны действительные числа a, b, c. Найдите среднее геометрическое их модулей. Формула среднего геометрического = $\left(\text{произведение чисел}\right)^{\frac{1}{\text{количество чисел}}}$.
2.	Дано вещественное число x. Составьте программу вычисления значения y по формуле $y = 4x^2 + 2(x^2 + 1) - \text{arctg}^3 x^2$.
	Вариант 11
1.	Даны катеты прямоугольного треугольника. Найдите его гипотенузу и площадь. Формулы: $c^2 = a^2 + b^2$, $S = \dfrac{ab}{2}$.
2.	Составьте программу вычисления значения y по формуле $y = \lvert 3(x + 1)^2 + 2(x + 1) + 2\rvert - \cos x^2$.
	Вариант 12
1.	Даны действительные положительные числа a, b, c, d, e. Найдите среднее геометрическое этих чисел. Формула среднего геометрического = $\left(\text{произведение чисел}\right)^{\frac{1}{\text{количество чисел}}}$.
2.	Дано вещественное число x. Составьте программу вычисления значения y по формуле $y = \cos(x^2 + 2x + 3) + \sin(x - 1)$.
	Вариант 13
1.	Даны действительные числа a, b, c, d. Найдите среднее арифметическое их модулей. Формула среднего арифметического = $\dfrac{\text{сумма чисел}}{\text{количество чисел}}$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \dfrac{x}{2} + \left(\dfrac{x}{2}\right)^2 + \left(\dfrac{x}{2}\right)^3 + \left(\dfrac{x}{2}\right)^4$.
	Вариант 14
1.	Дана сторона равностороннего треугольника. Найдите площадь этого треугольника. Формула $S = \dfrac{a^2\sqrt{3}}{4}$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \dfrac{x+1}{5} + (x+1)^2 + \sqrt[3]{x+1}$.

	Вариант 15
1.	Определить силу притяжения F между телами массы m_1 и m_2, находящимися на расстоянии r друг от друга. Формулы $F = \gamma \cdot \frac{m_1 \cdot m_2}{r}$, $\gamma = 6{,}6732 \cdot 10^{-11} \frac{\text{Нм}^2}{\text{кг}^2}$.
2.	Составьте программу вычисления значения y по формуле $y = \frac{x}{3} + \left(\frac{x}{3}\right)^2 + \sqrt[\frac{x}{3}]{\sin x} - 1$.
	Вариант 16
1.	Дана гипотенуза и катет прямоугольного треугольника. Найдите второй катет треугольника и радиус описанной окружности. Формулы: $c^2 = a^2 + b^2$, $R = \frac{c}{2}$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \frac{(x+1)^2 + 2(x+1)}{4} + \arcsin^2 \frac{x}{x+1}$.
	Вариант 17
1.	Даны гипотенуза и катет прямоугольного треугольника. Найдите радиус вписанной окружности. Формулы: $c^2 = a^2 + b^2$, $p = \frac{a+b+c}{2}$, $S = \sqrt{p(p-a)(p-b)(p-c)}$, $S = pr$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \frac{x^2}{2} + \left(\frac{x^2}{2}\right)^2 - \left(\frac{x^2}{2}\right)^4 + 3$.

	Вариант 18
1.	Известна длина окружности. Найдите площадь круга, ограниченного этой окружностью. Формулы: $S = \pi r^2$, $l = 2\pi r$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \frac{x+1}{3} + 2(x+1)^2 + \frac{3}{4}(x+1)^3$.
	Вариант 19
1.	Найдите площадь кольца, внутренний радиус которого равен 20, а внешний – заданному числу r ($r > 20$). Формула: $S = \pi r^2$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \frac{x+1}{3} + (x+1)^2 - \sin^2(x+1)^3$.
	Вариант 20
1.	Определить расстояние, которое пройдет равноускоренно движущееся тело, если известны ее начальная скорость, ускорение и время движения. Формула: $S = v_0 t + \frac{at^2}{2}$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \frac{x^2+1}{2} - \frac{27}{x^2} - \operatorname{tg} \frac{1}{x^2+1}$.
	Вариант 21

1.	Найдите сумму членов арифметической прогрессии a, $a + d$, …, $a + (n - 1) \cdot d$ по данным значениям a, d, n. Формула: $S_n = \frac{2a_1 + d(n-1)}{2} n$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = (x^2 + x - 2)(x^2 + x - 3) - 12(x^2 + x)$.
	Вариант 22
1.	Найдите площадь равнобочной трапеции с основаниями a и b и углом α при большем основании a. Формулы: $h = \frac{b-a}{2}\operatorname{tg}\alpha$, $S = \frac{a+b}{2}h$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = (12x - 1)(6x - 1) - 5\sin^2(3x - 1)^2$.
	Вариант 23
1.	Треугольник задан величинами своих углов и радиусом описанной окружности. Найдите стороны треугольника. Формула: $\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma} = 2R$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = (x + 2) - \operatorname{arctg}(x + 8) \cdot (x + 12) - 4x^2$.
	Вариант 24
1.	Треугольник задан длинами сторон. Найдите длины высот. Формулы: $p = \frac{a+b+c}{2}$, $h_a = \frac{2\sqrt{p(p-a)(p-b)(p-c)}}{a}$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = \frac{x}{x+3} + \frac{4}{x+1} - 2(x+4)$.
	Вариант 25
1.	Треугольник задан длинами сторон. Найдите длины медиан. Формула: $m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$.
2.	Составьте программу вычисления значения y по формуле $y = (6 - x)^4 + (8 - x)^4 - 16(10 - x)^4$.
	Вариант 26
1.	Треугольник задан длинами сторон. Найдите длины биссектрис. Формулы: $l_a = \frac{2ab\cos\frac{\gamma}{2}}{a+b}$, $c^2 = a^2 + b^2 - 2ab\cos\gamma$.
2.	Составьте программу вычисления значения y по формуле $y = 2x^2 + x^3 - \operatorname{arctg}(11x^2 + x + 2)$.
	Вариант 27
1.	Треугольник задан длинами сторон. Найдите радиус вписанной окружности. Формулы: $p = \frac{a+b+c}{2}$, $S = \sqrt{p(p-a)(p-b)(p-c)}$, $S = pr$.
2.	Дано действительное число x. Составьте программу вычисления значения y по формуле $y = (11x^2 - 6x - 9)^2 + x^3 + \sqrt[3]{x+2}$.
	Вариант 28
1.	Треугольник задан длинами сторон. Найдите радиус описанной окружности.

	Формулы: $S=\frac{abc}{4R}$, $p=\frac{a+b+c}{2}$, $S=\sqrt{p(p-a)(p-b)(p-c)}$.
2.	Дано действительное число *x*. Составьте программу вычисления *y* по формуле $y = x^3 + 2x^2 - x - \text{tg arcsin}\ \frac{x^2}{x^2+1}$.
	Вариант 29
1.	Вычислить расстояние между точками *A* и *B* с координатами x_1, y_1, z_1 и x_2, y_2, z_2. Формула: $\lvert AB\rvert=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$.
2.	Дано действительное число *x*. Составьте программу вычисления *y* по формуле $y=\frac{1}{x-1}-\frac{2}{x+2}-3(x+4)^2$.
	Вариант 30
1.	Треугольник в пространстве задан координатами своих вершин. Найдите периметр треугольника. Формула: $\lvert AB\rvert=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$.
2.	Дано действительное число *x*. Составьте программу вычисления *y* по формуле $y=\text{ctg}\left(\frac{1}{x}+\frac{1}{x+2}+\frac{1}{x+5}+\frac{1}{x+9}\right)$.

Решение типового варианта

Пусть вариант студента 31 содержит следующие задания:

	Вариант 31
1.	Октаэдр задан своим ребром *a*. Найдите объем *V* и площадь боковой поверхности *S* октаэдра. Формулы: $V=\frac{a^3\sqrt{2}}{3}$, $S=2a^2\sqrt{3}$.
2.	Дано действительное число *x*. Составьте программу вычисления *y* по формуле $y=\arcsin\left(\frac{x}{x+1}-\frac{x}{x+3}+\frac{x}{x+6}-\frac{x}{x+10}\right)$.

Сформируем отчет к лабораторной работе 2.

1. Титульный лист.

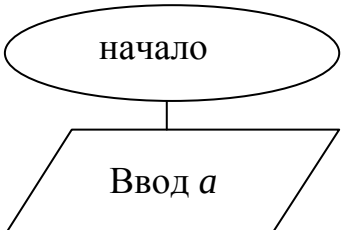
2. Сформулируем задание варианта 31.

1. Октаэдр задан своим ребром *a*. Найдите объем *V* и площадь боковой поверхности *S* октаэдра. Формулы: $V=\frac{a^3\sqrt{2}}{3}$, $S=2a^2\sqrt{3}$.

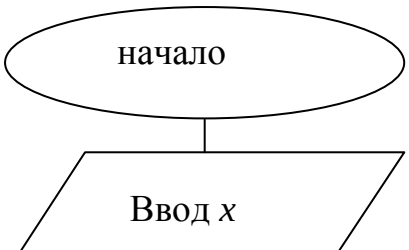
2. Дано действительное число *x*. Составьте программу вычисления *y* по формуле $y=\arcsin\left(\frac{x}{x+1}-\frac{x}{x+3}+\frac{x}{x+6}-\frac{x}{x+10}\right)$.

3. Составим блок-схему алгоритмов заданий варианта.

1.



2.



4. Напишем программу по составленной блок-схеме.

```
1. Uses crt;
    Var a, v, s: real;
    BEGIN
     ClrScr;
     Write (‘Введите a:’);
     Readln (a);
         V := sqr (a) * a * sqrt(2)/3;
     S := 2 * sqr (a)*sqrt(3);
     Writeln (‘V=’, V: 6: 3);
     Writeln (‘S=’, S: 6: 3);
     Readkey
    END.

2.  Uses crt;
    Var x, y, a, b: real;
    BEGIN
     ClrScr;
```

```
    Write ('Введите x:');
    Readln (x);
    A:= x + 6;
    B:= x/(a-5) – x/(a – 3) + x/a – x/(a + 4);
    Y := arctan (b/sqrt(1-sqr(b)));
    Writeln ('Y=', y: 6: 3);
    Readkey
  END.
```

5. Подберем значения для тестирования программ.

1. A = 1, тогда V = 0.471; S = 3.464;

A = 3, тогда V = 12.728; S = 31.177.

2. X = 1, тогда Y = 0.307;

X = 3, тогда Y = 0.360.

6. В текстовом редакторе WORD оформим отчет в соответствии с содержанием.

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1. Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

3. Новичков, В. С. Паскаль : учеб. пособие для сред. спец. учеб. заведений / В. С. Новичков, Н. И. Парфилова, А. Н. Пылькин. – М. : Высш. шк., 1990. – 223 с. : ил. – (Алгоритмические языки в техникуме).

4. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV.

Вопросы для самопроверки

1. Укажите назначение переменных в программе на языке Паскаль. Приведите пример объявления переменных *a* – вещественного типа, *b* – целочисленного типа, *c* – строкового типа.

2. Перечислите некоторые простые типы данных в Паскале (не менее двух). Приведите пример объявления переменных *a* – вещественного типа, *b* – целочисленного типа, *c* – переменной для хранения значения одного символа.

3. Перечислите некоторые целочисленные типы данных в Паскале (не менее двух). Приведите пример объявления переменных *a*, *b* – разных целочисленных типов так, чтобы переменная *a* могла принимать значения от 1 до 100, *b* – значения от –100 000 до 100 000.

4. Укажите вещественный тип данных в Паскале (можно один). Приведите пример объявления переменной *a* вещественного типа так, чтобы она могла принимать значение 2,7.

5. Укажите строковый тип данных в Паскале. Приведите пример объявления переменной *a* строкового типа так, чтобы она могла принимать значение ‘PASCAL 7.0’.

6. Укажите логический тип данных в Паскале. Приведите пример объявления переменной *a* логического типа.

7. Опишите последовательность действий при выполнении оператора присваивания. Приведите условное обозначение оператора присваивания.

8. В чем различие и в чем сходство данных типа CHAR и STRING?

9. Из каких разделов состоит программа на Турбо Паскале?

10. Назовите один обязательный раздел для работы программы на Турбо Паскале.

11. Назовите один необязательный раздел для работы программы на Турбо Паскале.

12. Переменные *a* и *b* описаны следующим образом A: INTEGER; B: STRING[5]; Предскажите, как выполнится последовательность операторов A:=’proba’; B:=’abrakadabra’.

13. Переменные *a* и *b* описаны следующим образом A: CHAR; B: STRING[5]; Предскажите, как выполнится последовательность операторов A:=’proba’; B:=’abrakadabra’.

14. Переменные *a* и *b* описаны следующим образом A: INTEGER; B: REAL; Предскажите, как выполнится оператор A:= B.

15. Переменные *a* и *b* описаны следующим образом A: INTEGER; B: REAL; Предскажите, как выполнится оператор B:= A.

ЛАБОРАТОРНАЯ РАБОТА 3. ГРАФИЧЕСКИЙ РЕЖИМ КОМПЬЮТЕРА ГРАФИЧЕСКИЕ ОПЕРАТОРЫ TURBO PASCAL 7.0 (6 ЧАСОВ)

Цель работы: освоить графические операторы TURBO PASCAL 7.0. Приобрести навыки работы с простейшей структурой алгоритмических языков программирования – линейной.

Теоретические положения

Графический режим

До сих пор основными объектами, появляющимися на экране, были символы. С их помощью можно в случае надобности построить изображение, однако более естественно использовать для этой цели графические операторы, которые позволяют создавать на экране геометрические фигуры (точки, отрезки, окружности). Все графические процедуры и функции Турбо Паскаля содержатся в модуле **GRAPH**, который следует подключить командой **USES.**

Компьютер может оперировать либо с символами (символьный режим), либо с геометрическими фигурами (графический режим). Монитор может работать в одном из пяти графических режимов. Для переключения экранных графических режимов используется процедура **INITGRAPH**, после которой располагаются графические операторы. Чтобы закрыть графический режим и вернуться в текстовый, используется процедура **CLOSEGRAPH**. Перед процедурой CloseGraph следует разместить какой-либо оператор задержки

(READLN, READKEY), иначе изображение исчезнет раньше, чем удастся его увидеть.

Процедура **InitGraph** имеет три параметра, которые позволяют выбирать наиболее подходящий режим работы. Для первых двух мы заведем переменные:

1. grDriver – переменная, в которую необходимо записать код требуемого графического драйвера (графические драйверы представляют собой файлы с расширением .BGI, которые обеспечивают взаимодействие программ с графическими устройствами). Нами будут использоваться EGA (численное значение этой предопределенной переменной равно 3) и VGA (численное значение равно 9). Если присвоить этой переменной значение DETECT, то произойдет автоматическое определение возможных значений.

2. grMode – переменная, в которую процедура помещает код графического режима.

Таблица 1

Адаптор	**Предопределенная константа**	**Численное значение**	**Резрешение**	**Цветность**
EGA	EGALo	0	640 x 200 пикселей	16 цветов
	EGAHi	1	640 x 350 пикселей	16 цветов
VGA	VGALo	0	640 x 200 пикселей	16 цветов
	VGAMed	1	640 x 350 пикселей	16 цветов
	VGAHi	2	640 x 480 пикселей	16 цветов

3. DriverPath – строка, содержащая путь к драйверу. В кабинете ВТ для загрузки графического режима можно установить путь к графическому драйверу при помощи последовательности действий, которую следует выполнить после запуска системы Турбо Паскаля. Действия выполняются в следующем порядке:

Функциональная клавиша F10 → Опция FILE → Опция ENTER → Опция CHANGE DIR.

Появится диалоговое окно, в котором рабочая строка ввода: DIRECTORY NAME и диалоговое окно DIRECTORY TREE, а также набор кнопок (Ok, Chdir, Revert, Help). Сразу при открытии активным становится содержимое строки DIRECTORY NAME. Клавишей TAB перейти в окно DIRECTORY TREE. DIRECTORY TREE – это окно, в котором приводится дерево каталогов текущего диска. Перемещение по дереву осуществляется клавишами ↓ ↑. Сделать активным путь TP7\BIN. Нажимайте TAB, пока активным не станет кнопка Ok. Нажмите ENTER – и можно работать с графикой. Процедура InitGraph с такими установками находит драйвер, если будет установлена просто пустая строка (‘’ – два рядом стоящих апострофа). Другой способ используется чаще, так как он позволяет при минимальных изменениях увидеть результат работы программы. Процедура InitGraph может содержать в параметрах реальный путь к драйверу, например, так: (grDriver, grMode,'g:\tp7\bgi');

CloseGraph служит для удаления драйвера из памяти. Правила хорошего программирования требуют, чтобы по окончании работы программы система вернулась в состояние, идентичное состоянию ее до запуска, поэтому графический режим желательно сохранять на экране только тот период времени, пока программа работает.

Следует помнить, что нумерация точек идет от левого верхнего угла с нуля. Номер строки соответствует координате *Y*, а номер точки в строке – координате *X*.

Очистка графического экрана

Процедура **CLEARDEVICE.**

Очистка графического экрана не очень востребована в малых учебных программах, так как переход из текстового экрана в графический и обратно очищает любой экран (текстовый или графический) сам собой в любом случае. Несколько изображений легко размещает-

ся на одном графическом экране. Поэтому очищать экран возникает необходимость крайне редко.

Формат процедуры без параметров.

ClearDevice; – очищает графический экран, закрашивает его в цвет фона, устанавливает указатель текущей позиции в точку с координатами (0, 0), то есть в верхний левый угол экрана. Координата *X* возрастает вправо, координата *Y* возрастает вниз. Цвет фона задается процедурой SetBkColor.

Построение контуров фигур

Чтобы задать цвет контура следует использовать процедуру **SETCOLOR**(*Color*). В скобках указывается цвет контура.

Любой контур можно нарисовать линиями различных типов. Для этого служит процедура **SETLINESTYLE** (*LineStyle, Pattern, Trickness*), где параметры:

LineStyle – определяет тип линии. На этом месте может стоять константа из следующей таблицы или соответствующее этой константе выражение.

Таблица 2

Константа	Значение	Описание
SolidLn	0	Сплошная
DottedLn	1	Пунктирная
CenterLn	2	Штрих пунктирная
DashedLn	3	Штриховая

Pattern – задает шаблон линии, начинающим программистам лучше всего брать ее всегда равной нулю.

Trickness – устанавливает толщину линий, может принимать значения перечисленных ниже констант:

Таблица 3

Константа	Значение	Описание
NormWidth	1	Нормальной толщины

ThickWidth	3	Удвоенной толщины

Процедура SetLineStyle ставится перед графическим оператором и действует на все операторы построения контуров изображения до повторного использования процедуры SetLineStyle.

Обе ниже расположенные программы рисуют линию из верхнего левого угла экрана в точку с координатами (100, 120). Определите из предыдущего материала общее количество точек в каждом из этих графических режимов.

1.
```
Uses Graph, Crt;
  Var
     grDriver: Integer;
     grMode: Integer;
  BEGIN
     grDriver := EGA;
     grMode := EGAHi;
     InitGraph (grDriver, grMode,' ');
     Line (0, 0, 100, 120);
     Readkey;
     CloseGraph;
  END.
```

2.
```
Uses Graph, Crt;
Var
     grDriver: Integer;
     grMode: Integer;
  BEGIN
     grDriver := DETECT;
     InitGraph (grDriver, grMode,' ');
     Line(0, 0, 100, 120);
     Readkey;
     CloseGraph;
  END.
```

Вариант под номером 2, дает обычно большее количество точек на экране, а значит, изображение иного качества. Компьютеров, для которых изображение одинаково, незначительное количество. Если изображение разное для этих двух программ, желательно в дальнейшем использовать именно тот графический режим, который дает изображение, лучшее для глаз. Если при запуске появляется сообщение об ошибке, следует попробовать заменить рядом стоящие апострофы (‘ ‘) на реальный путь к графическому драйверу, который вам сообщит специалист, обслуживающий Ваш компьютер. Указать путь для любого случая расположения графического драйвера не представляется возможным, так как Турбо Паскаль может грузиться с сервера, который для конкретного кабинета ВТ может иметь разное имя. Этот путь может быть G:\TP7\BGI, где G: – имя сервера для данного персонального компьютера, TP7\BGI – расположение графического драйвера на этом сервере.

Контуры элементарных фигур

Точка

PUTPIXEL(*x, y, color*) – закрашивает пиксель (элементарную точку экрана в графическом режиме) с заданными координатами (*x, y*) указанным цветом (*color*).

Отрезок

LINE(*x1, y1, x2, y2*) – рисует отрезок. Первые две координаты (*x1, y1*) – начало этого отрезка, две оставшиеся (*x2, y2*) – конец отрезка.

Прямоугольник

RECTANGLE(*x1, y1, x2, y2*) строит контур прямоугольника со сторонами параллельными краям экрана. (*x1, y1*) и (*x2, y2*) – координаты одной из диагоналей прямоугольника.

Окружность

CIRCLE(*x, y, radius*); – строит окружность, с центром в точке (*x, y*). *radius* – целочисленное выражение или константа.

Дуга окружности

ARC(*x, y, nd, kd, radius*) – строит дугу окружности, (*x, y*) – координаты центра дуги, *nd* – угол в градусах до начальной точки дуги, отсчитываемый против часовой стрелки от горизонтальной оси, направленной слева на право, *kd* – угол в градусах до конечной точки дуги, отсчитываемый против часовой стрелки от горизонтальной оси, направленной слева на право, *radius* – радиус дуги.

Угол задается в градусах (от 0 до 360).

Эллипс и дуга эллипса

ELLIPSE(*x, y, nd, kd, xr, yr*) – рисует дугу эллипса, (*x, y*) – координаты центра дуги эллипса, *nd* – угол в градусах до начальной точки дуги, отсчитываемый против часовой стрелки от горизонтальной оси, направленной слева направо, *kd* – угол в градусах до конечной точки дуги, отсчитываемый против часовой стрелки от горизонтальной оси, направленной слева направо, *xr* и *yr* – вертикальная и горизонтальная полуоси эллипса.

Задав *nd* и *kd* соответственно значения 0 и 360, получим изображение эллипса.

Построение закрашенных изображений

В Турбо Паскале заполненные изображения строятся в два этапа:

1 этап. Установить цвет и тип заполнения: процедура **SETFILLSTYLE**(*Pattern, Color*).

Типы заполнения могут быть следующими, в зависимости от значения переменной или константы, стоящей на месте переменной *Pattern*. На этом месте может стоять константа из следующей таблицы или соответствующее этой константе выражение.

Таблица 4

Константа	Значение	Узор
EmptyFill	0	Сплошной цветом фона
SolidFill	1	Сплошной текущим цветом
LineFill	2	Заполнение горизонтальными линиями
LtSlashFill	3	Типа / / нормальной толщины
SlashFill	4	Типа / / удвоенной толщины
BkSlashFill	5	Типа \ \ удвоенной толщины
LtBkSlashFill	6	Типа \ \ нормальной толщины
HatchFill	7	Заполнение клеткой
XhatchFill	8	Заполнение косой редкой клеткой
InterleaveFill	9	Заполнение косой частой клеткой
WideDotFill	10	Заполнение редкими точками
CloseDotFill	11	Заполнение частыми точками

Цвет определяется обычным набором констант (см. лабораторную работу 1).

2 этап. Построить изображение.

Список элементарных закрашенных изображений с описанием параметров приводится ниже

Закрашенный прямоугольник

BAR(*x1, y1, x2, y2*) – строит закрашенный прямоугольник со сторонами, параллельными краям экрана, (*x1, y1*) и (*x2, y2*) – координаты одной из диагоналей прямоугольника.

Закрашенный эллипс

FILLELLIPSE(*x, y, xr, yr*) – строит «залитый» ранее определенным цветом и узором эллипс, *x, y* – координаты центра эллипса, *xr, yr* – горизонтальная и вертикальная оси закрашенного эллипса.

Закрашенный сектор эллипса

SECTOR(*x, y, nd, kd, xr, yr*) – строит закрашенный сектор эллипса, *x, y* – координаты центра эллипса, *nd, kd* – начало и конец дуги в градусах, *xr, yr* – горизонтальная и вертикальная оси.

Закрашенный сектор окружности

PIESLICE(*x, y, nd, kd, radius*) – строит закрашенный сектор окружности, *x, y* – координаты центра окружности, *nd, kd* – начало и конец дуги в градусах, *radius* – радиус окружности.

Закрашивание замкнутой области

FLOODFILL(*x, y, Border*) – заполняет заданным с помощью SetFillStyle стилем области, расположенной либо внутри замкнутого контура, либо вне него, *x, y* – координаты точки внутри или вне замкнутого контура, от которой распространяется заливка, *Border* – цвет контура замкнутой области.

Область не должна иметь разрывов (то есть граничные точки соприкасаются). Исходная точка не должна лежать на границе.

В теоретических положениях – список операторов, достаточный для выполнения любых заданий лабораторных работ. Желающим самостоятельно расширить свои познания по созданию в Турбо Паскале графических изображений можно изучить учебные пособия из списка литературы.

Порядок выполнения работы

1. Уточните номер своего варианта.

2. Ознакомьтесь со списком графических операторов в теоретических положениях.

3. Составьте список операторов, необходимых для исполнения Вашего изображения.

4. Напишите программу, строящую заданное в варианте изображение. По желанию, рисунок может быть негативом, (то есть черные места сделать белыми, а белые сделать черными), хотя бы один из рисунков следует сделать трехцветным. При написании программы можно пойти двумя путями. Во-первых, можно изобразить рисунок на миллиметровой бумаге, определить для себя оси координат и размеры экрана, затем указывать в операторах получающиеся коор-

динаты каждого элементарного оператора. Во-вторых, допустимо написать оператор с произвольными данными, а затем добиваться, чтобы элемент изображения встал на свое место. Второй способ часто гораздо эффективнее, но требует хорошего пространственного изображения.

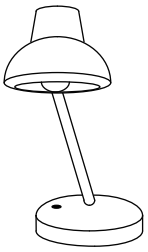
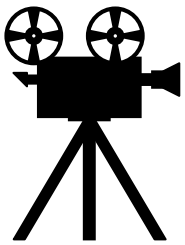
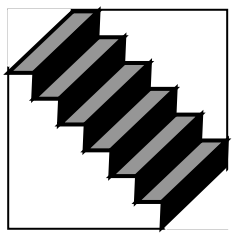
5. Отладьте программу (исключите все сообщения об ошибках).

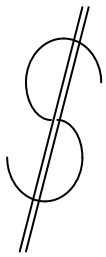
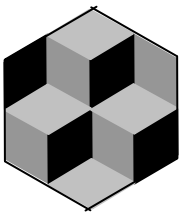
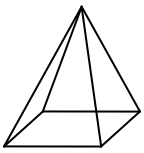
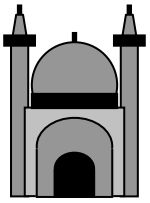
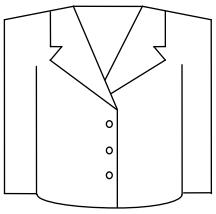
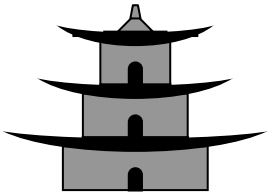
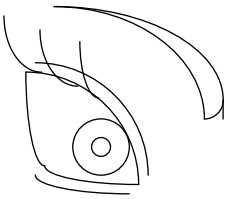
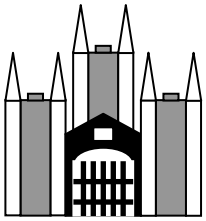
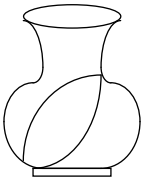
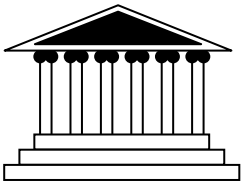
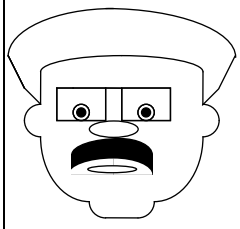
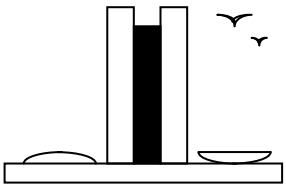
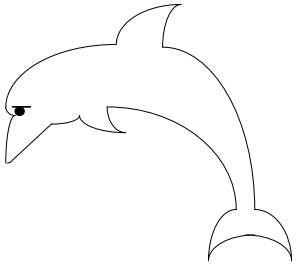
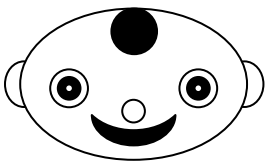
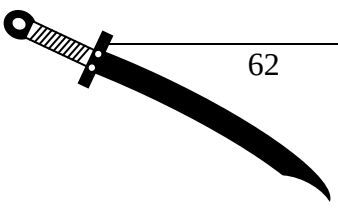
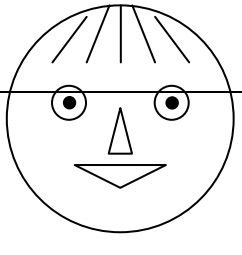
6. В текстовом редакторе WORD или рукописно оформите отчет в соответствии с содержанием.

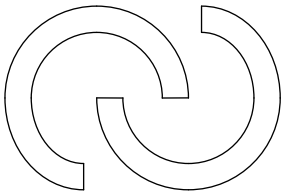
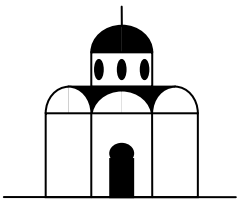
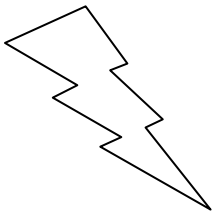
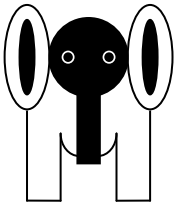
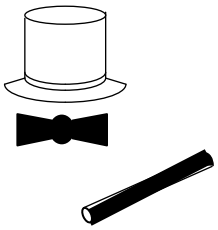
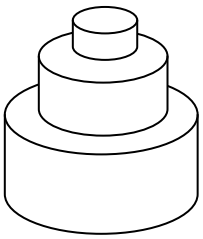
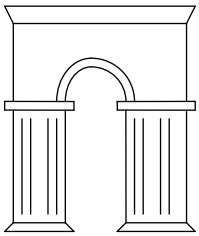
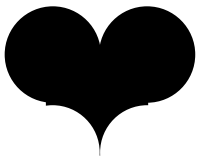
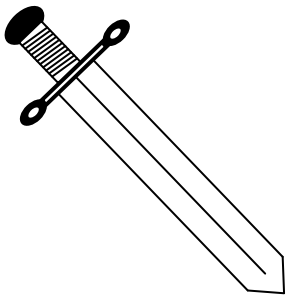
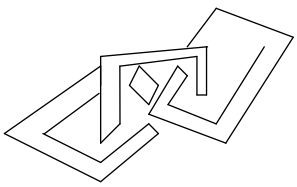
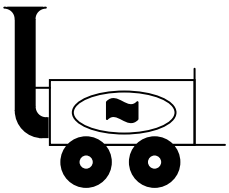
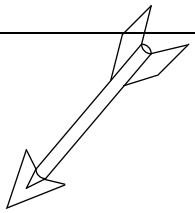
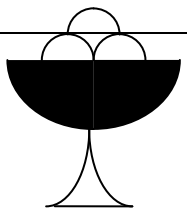
Содержание отчета

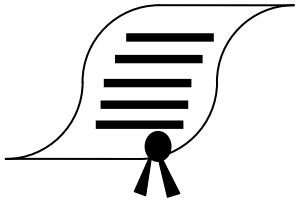
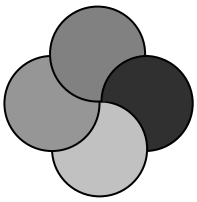
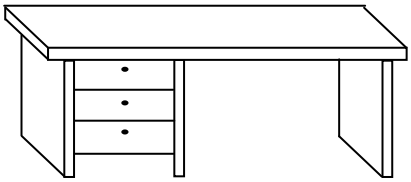
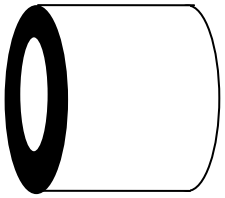
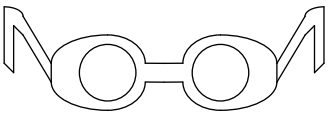
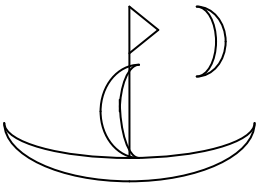
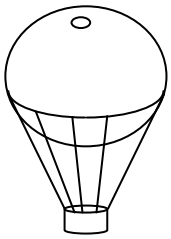
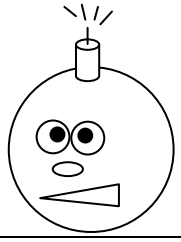
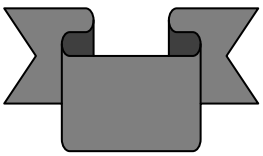
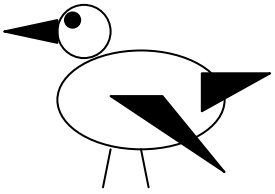
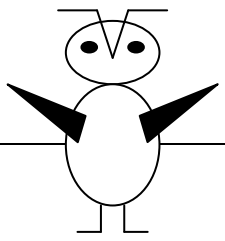
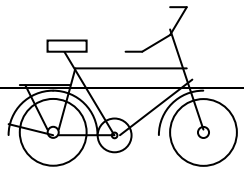
1. Титульный лист.
2. Задания варианта.
3. Описание использованных в работе операторов.
4. Программные единицы.

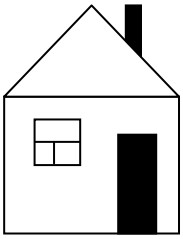
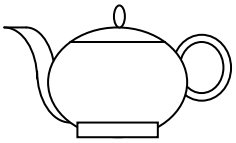
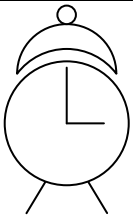
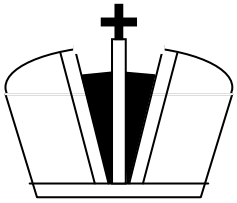
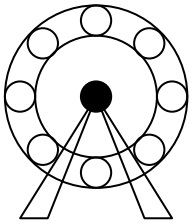
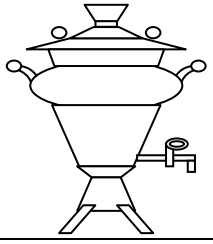
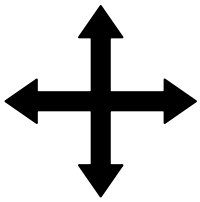
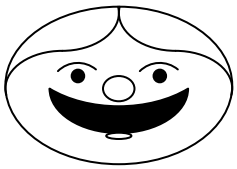
Варианты задач

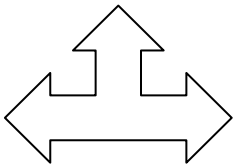
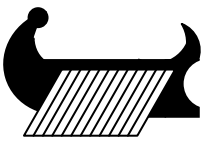
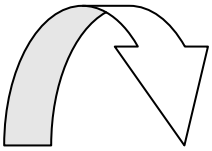
Вариант 1		
Вариант 2		

Вариант 3		
Вариант 4		
Вариант 5		
Вариант 6		
Вариант 7		
Вариант 8		
Вариант 9		
Вариант 10		

Вариант 11		
Вариант 12		
Вариант 13		
Вариант 14		
Вариант 15		
Вариант 16		
Вариант 17		

Вариант 18		
Вариант 19		
Вариант 20		
Вариант 21		
Вариант 22		
Вариант 23		
Вариант 24		

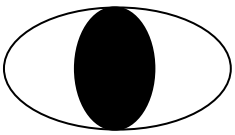
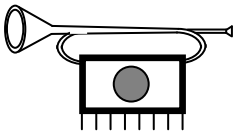
Вариант 25		
Вариант 26		
Вариант 27		
Вариант 28		

Вариант 29		
Вариант 30		

Решение типового варианта

Рассмотрим пример оформления работы.

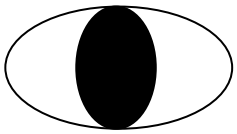
Пусть вариант студента 31 содержит следующие задания.

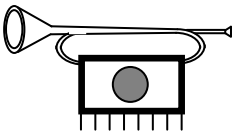
Вариант 31.		

Сформируем отчет к лабораторной работе 3.

1. Титульный лист.

2. Сформулируем задание варианта 31.

1. 

2. 

3. Описание использованных в работе операторов.

В программе будут использованы операторы.

Отрезок

LINE(*x1*, *y1*, *x2*, *y2*) – рисует отрезок. Первые две координаты (*x1*, *y1*) – начало этого отрезка, две оставшиеся (*x2*, *y2*) – конец отрезка.

Закрашенный эллипс

FILLELLIPSE(*x*, *y*, *xr*, *yr*) – строит «залитый» ранее определенным цветом и узором эллипс, *x*, *y* – координаты центра эллипса, *xr*, *yr* – горизонтальная и вертикальная оси закрашенного эллипса.

Прямоугольник

RECTANGLE(*x1*, *y1*, *x2*, *y2*) строит контур прямоугольника со сторонами, параллельными краям экрана. (*x1*, *y1*) и (*x2*, *y2*) – координаты одной из диагоналей прямоугольника.

4. Программные единицы.

```
{1.} uses crt, graph;
     Var Gd, Gm: Integer;
     Begin
       Gd := Detect;
       InitGraph (Gd, Gm, '');
```

```
      Ellipse (100, 100, 0, 360, 50, 30);
      FillEllipse(100, 100,30, 30);
      Readkey;
      CloseGraph;
    ISBN 5-8424-0356-0 ISBN 5-8424-0356-0
    End.
{2.} Uses crt, graph;
    Var Gd, Gm: Integer;
    Begin
      Gd := detect;
      InitGraph (Gd, Gm, '');
      Line (50,40,180,45);
      Line (50,55,180,50);
      Line (50,40,50,55);
      Line (50,40,30,33);
      Line (50,55,30,61);
      Line (180,45,180,50);
      Line (185,43,185,52);
      Line (180,45,185,43);
      Line (180,50,185,52);
      Ellipse (30, 47, 0, 360, 7, 14);
      Rectangle (55, 110, 150, 180);
      Ellipse (70, 84, 110, 240, 30, 30);
      Ellipse (75, 84, 110, 240, 30, 30);
      Ellipse (130, 84, 300, 85, 30, 30);
      Ellipse (135, 84, 300, 85, 30, 30);
     SetFillStyle (1, 4);
      FillEllipse (100, 145,30, 30);
      Line (60,180,60,210);
      Line (72,180,72,210);
      Line (84,180,84,210);
      Line (96,180,96,210);
      Line (108,180,108,210);
```

```
    Line (120,180,120,210);
    Line (132,180,132,210);
    Line (144,180,144,210);
    Readkey;
    CloseGraph;
  end.
```

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1. Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

3. Довгаль, С. И. Персональные ЭВМ : ТурбоПаскаль V 7.0. Объектное программирование. Локальные сети : учебное пособие / С. И. Довгаль, Б. Ю. Литвинов, А. И. Сбитнев.

4. Епанешников, А. Программирование в среде Turbo Pascal 7.0 / А. Епанешников, В. Епанешников. – М. : «ДИАЛОГ-МИФИ», 1993. – 288 с.

5. Турбо Паскаль 7.0 – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

6. Хершель, Рудольф. Турбо Паскаль / Рудольф Хершель. – 2-е изд., перераб., – Вологда : МП «МИК», 1991. – 342 с. при участии МП ТПО «Квадрат», г. Москва.

Вопросы для самопроверки

1. Формат процедуры работы с графикой PUTPIXEL выглядит следующим образом: **PROCEDURE PUTPIXEL (X, Y: INTEGER, COLOR: WORD);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

2. Формат процедуры работы с графикой ARC выглядит следующим образом: **PROCEDURE ARC (X, Y: INTEGER; STANGLE,**

ENDANGLE, RADIUS: WORD); Расшифруйте каждое слово этого формата. Укажите название фигуры.

3. Формат процедуры работы с графикой BAR выглядит следующим образом: **PROCEDURE BAR (X1, Y1, X2, Y2: INTEGER);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

4. Формат процедуры работы с графикой CIRCLE выглядит следующим образом: **PROCEDURE CIRCLE (X, Y: INTEGER; RADIUS: WORD);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

5. Формат процедуры работы с графикой ELLIPSE выглядит следующим образом: **PROCEDURE ELLIPSE (X, Y: INTEGER; STANGLE, ENDANGLE: WORD; XRADIUS, YRADIUS: WORD);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

6. Формат процедуры работы с графикой FILLELIIPS выглядит следующим образом: **PROCEDURE FILLELLIPSE (X, Y: INTEGER; XRADIUS, YRADIUS: WORD);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

7. Формат процедуры работы с графикой FLOODFILL выглядит следующим образом: **PROCEDURE FLOODFIL (X, Y: INTEGER; BORDER: WORD);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

8. Формат процедуры работы с графикой INITGRAPH выглядит следующим образом: **PROCEDURE INITGRAPH (VAR GRAPHDRIVER: INTEGER; VAR GRAPHMODE: INTEGER; DRIVERPATH: STRING);** Расшифруйте каждое слово этого формата. Укажите действие процедуры.

9. Формат процедуры работы с графикой LINE выглядит следующим образом: **PROCEDURE LINE (X1, Y1, X2, Y2: INTEGER);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

10. Формат процедуры работы с графикой PIESLICE выглядит следующим образом: **PROCEDURE PIESLICE (X, Y: INTEGER; STANGLE, ENDANGLE, RADIUS: WORD);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

11. Формат процедуры работы с графикой RECTANGLE выглядит следующим образом: **PROCEDURE RECTANGLE (X1, Y1, X2, Y2: INTEGER);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

12. Формат процедуры работы с графикой SECTOR выглядит следующим образом: **PROCEDURE SECTOR (X, У: INTEGER; STANGLE, ENDANGLE, XRADIUS, YRADIUS: WORD);** Расшифруйте каждое слово этого формата. Укажите название фигуры.

13. Сформулируйте отличие графического экрана от текстового экрана.

14. Изобразите на эскизе, что появится в результате выполнения в графическом режиме фрагмента с последовательностью операторов: **SETCOLOR (WHITE); RECTANGLE (10, 10, 100, 100); BAR (5, 5, 120, 120);** цвета закраски фигур действуют по умолчанию. Поясните, почему появится именно такая фигура.

15. Изобразите на эскизе, что появится в результате выполнения в графическом режиме фрагмента с последовательностью операторов: **SETCOLOR (WHITE); BAR (5, 5, 120, 120); RECTANGLE (10, 10, 100, 100);** цвета закраски фигур действуют по умолчанию. Поясните, почему появится именно такая фигура.

ЛАБОРАТОРНАЯ РАБОТА 4.
ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ И ОПЕРАТОРЫ ЦИКЛА (6 ЧАСОВ)

Цель работы: освоить способы реализации многократно повторяемых действий. Приобрести навыки работы со структурой управляющих конструкций алгоритмических языков программирования – циклической. Изучить циклические конструкции: цикл «пока» while, цикл «до» repeat и цикл с параметром for.

Теоретические положения

Составной и пустой оператор

Составной оператор – это последовательность произвольных операторов программы, заключенная в операторные скобки – зарезервированные слова BEGIN … END. Составные операторы – важный инструмент Турбо Паскаля, позволяющий обходить ограничение единственности для многих операторов. Фактически весь раздел операторов, заключенный в операторные скобки, представляет собой один составной оператор.

Пустой оператор не содержит никаких действий. Это понятие введено в Паскаль для упрощения синтаксиса. Например, при копировании строк часто точка с запятой располагается неудачно: перед служебным словом END или две точки с запятой подряд. Некоторые реализации стандартного Паскаля интерпретировали такое сочетание как ошибку. Турбо Паскаль позволяет игнорировать отсутствие действий как в тексте программы, так и при создании кода.

Составной и пустой оператор, как любой мощный инструмент, требует практических навыков, то, что они могут всплывать в программе непроизвольно, часто приводит к созданию трудно выявляемых логических ошибок. Большое количество таких ситуаций сформулировано в ВОПРОСАХ ДЛЯ САМОПРОВЕРКИ. Нам эти понятия нужны для формулировок в циклических конструкциях.

Операторы повторений

Если вычислительный процесс содержит многократные вычисления по одним и тем же математическим зависимостям, но для различных значений входящих в них величин (переменных), то его называют циклическим. Многократно повторяемые участки вычислений называют циклами, а переменные, изменяющиеся в цикле, – переменными цикла. Алгоритм циклической структуры в наиболее общем виде должен содержать:

1) подготовку цикла: задание начальных значений переменным цикла перед первым его выполнением;

2) тело цикла: действия, повторяемые в цикле для различных значений переменных цикла;

3) модификацию (изменение) значений переменных цикла перед каждым новым его повторением;

4) управление циклом: проверку условия продолжения (или окончания) цикла и переход на начало тела цикла, если выполняется условие продолжения цикла (или выход из цикла по его окончании).

Цикл с параметром

Самый очевидный способ задать закон повторения реализован в цикле с параметром. Циклы такого типа называют также циклами со счетчиком. Число повторений тела цикла в этом случае подсчитывается с помощью специальной переменной (счетчика), для которой известны начальное и конечное (пороговое) значения, шаг ее изменения в Паскале всегда 1 или –1. Управление циклом осуществляется на основании сравнения текущего значения счетчика с заданным порогом. Счетчик также именуют параметром цикла.

Для программирования циклов с известным числом повторений существует оператор цикла с параметром. Этот оператор предусматривает повторное выполнение некоторого другого оператора с одновременным изменением по правилу арифметической прогрессии значения, присваиваемого управляющей переменной (параметру) этого цикла.

Оператор ***цикла с параметром*** имеет следующий вид:

FOR I :=A **TO** B **DO** P; (Шаг равен 1), или

FOR I :=A **DOWNTO** B **DO** P; (Шаг равен –1),

где FOR (ДЛЯ), TO (ДО), DOWNTO (ВНИЗ ДО), DO (ВЫПОЛНИТЬ) – служебные слова;

I – параметр цикла – переменная любого скалярного типа, кроме вещественного;

A, B – скалярные выражения, тип которых должен совпадать с типом переменной параметра цикла;
P – любой оператор языка Паскаль.

Оператор P выполняется в цикле FOR для каждого значения переменной параметра цикла начиная со значения A (начального) до значения B (конечного) включительно. При использовании в цикле служебного слова TO значение параметра цикла увеличивается, при DOWNTO – уменьшается. Шаг изменения параметра цикла зависит от типа этой переменной. Наиболее часто используется переменная целого типа (BYTE, INTEGER), что определяет шаг, равный 1 при TO, и –1 при DOWNTO.

В блок-схемах для схематического представления цикла с параметром, согласно стандарту, строят обычный цикл с предусловием (см. ниже) или используют специальный блок заголовка цикла (блок модификации), внутри которого указывают закон изменения параметра цикла. Структурограммы в разных источниках реализованы тоже по-разному. Воспользуемся представлением, максимально близким к школьному. Введем условные обозначения: НЦ – начальное значение параметра цикла, КЦ – конечное значение параметра цикла, ТЦ – тело цикла.

Схематические иллюстрации выполнения оператора FOR:

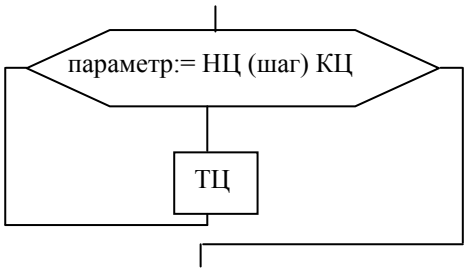


Рис. 1. Блок-схема

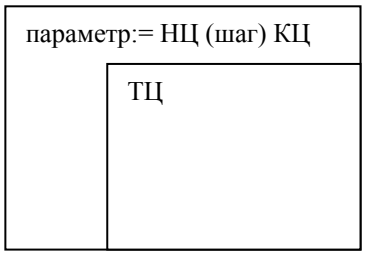


Рис. 2. Структурограмма по Насси и Шнейдерману

При программировании циклов с параметром необходимо помнить следующие правила организации цикла:

1) параметр цикла, начальное значение и конечное должны быть одинакового типа, их тип может быть любым скалярным типом (стандартным, перечисляемым, ограниченным), кроме вещественного;

2) очередное значение параметра цикла вычисляется автоматически по возрастанию в сочетании с TO или по убыванию в сочетании с DOWNTO; в частности, для целого типа шаг изменения значения параметра цикла равен 1 при TO и –1 при DOWNTO;

3) не следует изменять внутри (в теле) цикла значения переменной I выражений, A, B;

4) запрещено входить в цикл с помощью оператора GOTO, минуя оператор FOR, так как значения I, A, B будут не определены;

5) цикл не выполняется вообще, если начальное значение больше (при DOWNTO – меньше), чем конечное;

6) после служебного слова DO может стоять только один оператор; если в цикле нужно выполнить группу операторов, то их заключают в операторные скобки BEGIN-END.

Выше были рассмотрены алгоритмы циклической структуры с заранее известным числом повторений цикла. Достаточно часто приходится сталкиваться с циклическими вычислительными процессами, когда число повторений цикла неизвестно, а задано некоторое условие его окончания (или продолжения). Для программной реализации таких вычислительных процессов в языке Паскаль существует два типа операторов: оператор цикла с предусловием и оператор цикла с постусловием.

Цикл с предусловием. Оператор цикла с предусловием имеет следующую общую форму записи:

WHILE B **DO** P;

где WHILE (ПОКА), DO (ВЫПОЛНИТЬ) – служебные слова;

B – логическое выражение;

P – любой простой или составной оператор языка.

Схематические иллюстрации выполнения оператора WHILE

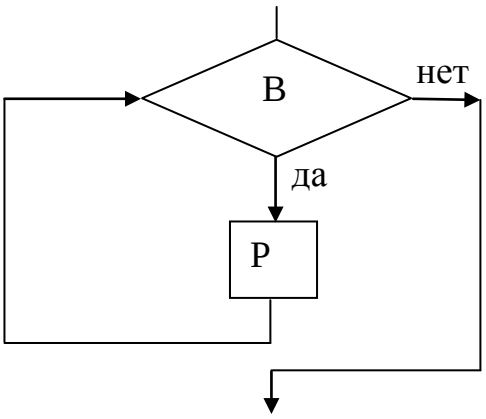


Рис. 3. На языке блок-схем.

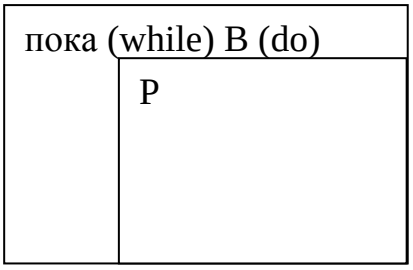


Рис. 4. В структурограммах по Насси и Шнейдерману

Оператор P выполняется в цикле WHILE до тех пор, пока условие B истинно (имеет значение TRUE). Если условие ложно (имеет значение FALSE), то выполняется оператор, следующий за WHILE. Если условие ложно с самого начала, то P не выполняется ни разу. Условие вычисляется и анализируется перед каждым выполнением цикла, отсюда и термин «предусловие».

Оператор цикла с постусловием. Оператор цикла с постусловием имеет следующий вид

REPEAT P **UNTIL** B;

где REPEAT (ПОВТОРЯТЬ), UNTIL (ДО ТЕХ ПОР, ПОКА) – служебные слова;

P – любое количество любых операторов языка;

B – логическое выражение.

Действие оператора REPEAT подобно действию оператора WHILE, но проверка условия производится после очередного цикла, что обеспечивает его выполнение хотя бы один раз. Служебные слова REPEAT UNTIL по действию похожи на операторные скобки BEGIN-END: между ними можно поместить группу операторов, отделяя их друг от друга точкой с запятой. Операторы выполняются в цикле REPEAT до тех пор, пока B ложно.

Оператор WHILE используется чаще оператора REPEAT. Это связано с тем, что во многих практических случаях желательно осуществлять проверку на окончание цикла до его выполнения и иметь возможность при необходимости пропустить цикл вообще.

При составлении циклов с предусловием или постусловием необходимо принимать во внимание следующие моменты:

1) перед каждым (первым) выполнением цикла условие его окончания (или продолжения) должно быть определено (иметь конкретное значение);

2) тело цикла должно содержать хотя бы один оператор, влияющий на B окончания (продолжения), иначе цикл будет продолжаться бесконечно;

3) условие B для окончания цикла должно быть выполнено;

4) условие B вычисляется при каждом выполнении цикла и поэтому должно быть максимально простым.

Схематические иллюстрации выполнения оператора REPEAT:

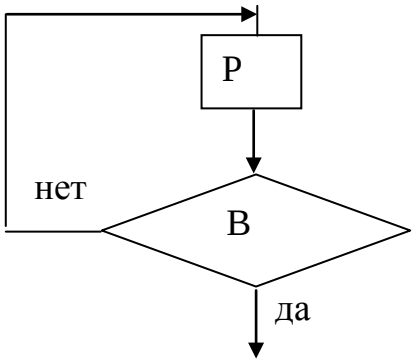


Рис. 5. На языке блок-схем

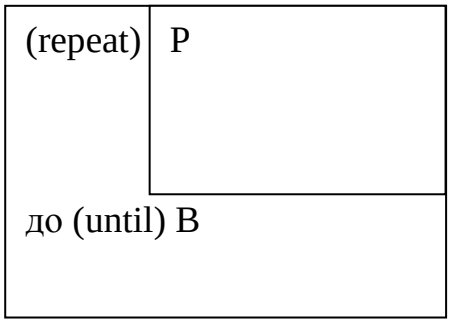


Рис. 6. В структурограммах по Насси и Шнейдерману

В структурограммах по Насси и Шнейдерману служебные слова в скобках – другой стандарт оформления, вполне допустимо использовать любой из форматов: со служебными словами на русском языке или на английском.

В блок-схеме, если она просто иллюстрация планируемых действий, допустимо не указывать конкретных значений, так как, если они не рассчитаны на бумаге, их все равно придется менять.

Опишем оператор цикла «**ПОКА**» с параллельным разбором примера с его составляющими.

Пример.

Построить пять концентрических окружностей с различными радиусами.

Таблица 1

Описание оператора цикла «ПОКА»	**Пример**
Оператор цикла применяется в тех случаях, когда по условию задачи нужно **многократно** выполнить одинаковые действия, различающиеся некоторой величиной	Пять окружностей с различными радиусами
Для оператора цикла выделяется изменяемая величина, для которой заводится переменная (имя этой величины)	Обозначим радиусы через **R**
Переменной задается начальное значение. Первая окружность будет построена радиусом 50	R := 50
Оператор цикла начинается со служебного слова **while** (пока), после которого указывается условие, выполнение которого обеспечивает повторение тела цикла. R <= 200 – условие. Как только значение переменной станет больше 200, программа выйдет из цикла. Значок «<=» – означает меньше или равно (или небольше). Затем **do** – служебное слово (выполнить)	**while** R <= 200 **do**
В теле цикла предусматривается оператор изменения переменной на некоторую величину – шаг. Радиус каждой следующей окружности больше предыдущей на 50	R := R + 50

Простые условия в программе на Паскале формируются при помощи значков:

< меньше, <= не больше,

> больше, >= не меньше.

Пример.

Выведем на экран квадраты нечетных чисел от 7 до 27. Легко заметить, что требуемые значения подряд идущих нечетных чисел можно получить следующим образом:

2 * 3 + 1 = 7,

2 * 4 + 1 = 9,

2 * 5 + 1 = 11,

2 * 6 + 1 = 13,

. . . ,

2 * 13 + 1 = 27,

и значит, программа может выглядеть так:

```
Var
   k: integer;
BEGIN
   For k := 3 to 13 do
      Writeln (sqr (2 * k + 1))
END.
```

Обратите внимание, что в теле цикла данной программы всего один оператор и операторные скобки нам не понадобились.

Если вместо служебного слова **TO** используется служебное слово **DOWNTO**, то параметр переберет значения по убыванию.

Пример.

FOR T **:=** 6 **DOWNTO** 1 **DO**, переменная T последовательно примет значения

6, 5, 4, 3, 2, 1.

Тот же результат можно получить оператором REPEAT.

Пример.

T := 6; REPEAT T := T – 1 UNTIL T <> 1, переменная T последовательно примет значения

6, 5, 4, 3, 2, 1.

Порядок выполнения работы

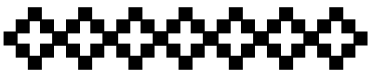
1. Уточните номер своего варианта.
2. Составьте блок-схему алгоритмов варианта.
3. Напишите программу по составленным блок-схемам.
4. Отладьте программы (исключите все сообщения об ошибках) и подберите значения для тестирования программ (запишите результаты тестирования).

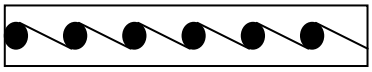
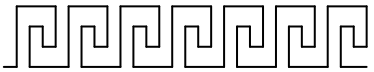
5. В текстовом редакторе WORD или рукописно создайте отчет в соответствии с содержанием.

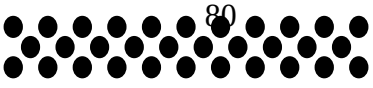
Содержание отчета

1. Титульный лист.
2. Задания варианта
3. Блок-схемы заданий 1 и 2.
4. Программные единицы.
5. Результат выполнения программы.

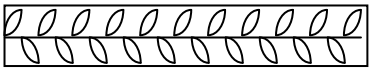
Варианты задач

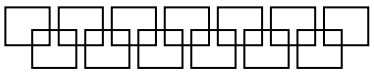
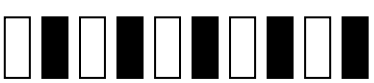
	Вариант 1
1.	Распечатать таблицу (с аргументами) значений функции на отрезке [0, 1] с шагом 0,1: $y = \sin x$.
2.	Напишите программу, которая выводит на экран компьютера картинку «звездного неба» со случайным набором точек, цвет тоже случаен, число точек ($\leq$ 5000) задается с клавиатуры.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 2
1.	Распечатать таблицу значений функции (с аргументами) на отрезке [0,1] с шагом 0,1: $y = \sqrt{x}$.
2.	Напишите программу, которая выводит на экран компьютера картинку мерцающего «звездного неба» со случайным набором точек, цвет тоже случаен, число точек ($\leq$ 5000) задается с клавиатуры.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 3
1.	Распечатать на экране квадраты четных чисел от 5 до 25.
2.	Напишите программу, которая выводит на экран компьютера картинку движущихся точек по карте «звездного неба», реализовать: случайный набор точек, цвет случайный, число точек ($\leq$ 250) задается с клавиатуры, движение точек реализовать по кругам с общим центром.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 4

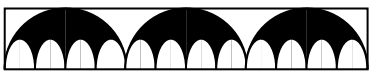
1.	Распечатать на экране квадраты нечетных чисел от 7 до 27.
2.	Напишите программу, которая выводит на экран компьютера картинку движущихся по прямой из левого верхнего в правый нижний угол «астероидов» – закрашенных кругов по карте «звездного неба», реализовать: случайный набор точек – «звезд» (≤ 100), цвет случайный, но восстанавливающийся после перекрытия «звезды» «астероидом», число «астероидов» (≤ 100), одновременно участвующих в движении по экрану задается с клавиатуры.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 5
1.	Распечатать на экране корни квадратные из чисел 3, 7, 11,..., 23.
2.	Пусть a_0=1, $a_k = k \cdot a_{k-1} + 1$, k = 1, 2, … . Дано натуральное число n. Получите a_n без использования массива или каких-либо других способов сохранения, где бы то ни было в памяти компьютера всех элементов этой последовательности значений.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 6
1.	Напечатайте на экране слово ПАСКАЛЬ лесенкой.
2.	Напишите программу без использования графики, которая выводит на экран компьютера картинку мерцающего «звездного неба» (то есть звезда пропадает и появляется в произвольном месте «звездного неба») со случайным набором точек (≤ 100, символ ‘*’), цвет символа тоже случаен, число точек задается с клавиатуры.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 7
1.	Используя оператор цикла, заставьте слово ПАСКАЛЬ двигаться по диагонали сверху вниз, из верхнего левого угла в правый нижний угол.
2.	Напишите программу с использованием графики, которая иллюстрирует решение математической задачи о движении двух путников, которые выходят из разных пунктов, находящихся на расстоянии 10 км навстречу друг другу и встречаются в определенной точке. Предусмотреть разметку частей расстояния. Путников изобразить в виде вертикальных отрезков.
3.	Используя оператор цикла с параметром, постройте орнамент.

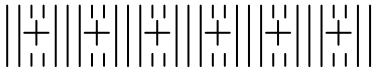


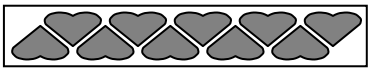
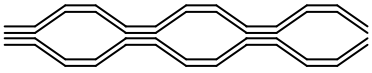
	Вариант 8
1.	Написать программу для нахождения суммы $1+2^2+3^2+\ldots+30^2$.
2.	Предположим, что Вы мечтаете стать миллионером. Это не трудно, просто для некоторых это занимает много времени. Надо только регулярно откладывать некоторые суммы денег, и, в конце концов, миллион рублей у Вас появится. Напишите программу, вычисляющую количество лет, которое Вам придется затратить, если годовой заработок у вас будет 30 000 руб. и Вы можете его весь перечислять в накопление.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 9
1.	Написать программу для нахождения суммы $\frac{1}{\sqrt{1}}+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{3}}+\ldots+\frac{1}{\sqrt{n}}$.
2.	Численно убедиться, является ли заданная функция $y = f(x)$ четной или нечетной на заданном отрезке $-a \le x \le a$. Учесть погрешность вычислений и возможные точки разрыва функции. Проверить, например, для функций $y = x^4$, $y = \operatorname{tg} x$, $y = e^x$, вычисляя их на отрезке [-5; 5] с шагом 0,1.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 10
1.	Написать программу для нахождения произведения $\sin 1^\circ \cdot \sin 2^\circ \cdot \sin 3^\circ \cdot \ldots \cdot \sin 15^\circ$.
2.	Утверждается, что функция $y = f(x)$ периодическая с периодом T. Проверить это численно, вычислив функцию с постоянным шагом на отрезке $[0; 5T]$. Учесть погрешность вычислений и возможные точки разрыва функций. Проверить на примере функций: $y = \sin^2 x$, $y = \operatorname{tg} x$ ($T = \pi$); $y = \frac{1}{x}\sin x$ ($T = 2\pi$).
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 11
1.	Написать программу для нахождения произведения $\frac{1}{1+m^2}\cdot\frac{1}{1+(m+1)^2}\cdot\ldots\cdot\frac{1}{1+n^2}$ при значениях $m < n$.
2.	Для заданных a и b найти все точки с целочисленными координатами, нахо-

	дящиеся внутри эллипса $\frac{x^2}{a^2}+\frac{y^2}{b^2}\le 1$. Используя процедуру **GotoXY** в Паскале, вывести найденные координаты точек на экран.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 12
1.	Найдите число *n*, для которого $\sqrt{1}+\sqrt{2}+\ldots+\sqrt{n}$ больше некоторого числа *k*.
2.	Определить количество членов ряда, необходимых для сходимости (число слагаемых для достижения заданной точности ε, то есть модуль очередного слагаемого меньше, чем заданная точность ε) следующего разложения числа π: $\pi = 4\cdot\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\ldots\right)$
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 13
1.	Дано натуральное число *n*. Вычислить 2^n.
2.	Предприниматель, начав дело, взял кредит размером *k* рублей под *p* процентов годовых и вложил его в свое дело. По прогнозам, его дело должно давать прибыль *r* рублей в год. Сможет ли он накопить сумму, достаточную для погашения кредита, и если да, то через сколько лет?
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 14
1.	Дано натуральное число *n*. Вычислить: $n!$.
2.	Каждая из деталей должна последовательно пройти обработку на каждом из трех станков. Продолжительности обработки каждой детали на каждом станке вводятся группами по 3 числа до исчерпания ввода. Сколько времени займет обработка всех деталей?
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 15
1.	Дано натуральное число *n*. Вычислить: $\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{n^2}\right)$.
2.	Найдите так называемое «машинное ипсилон» – такое минимальное, не равное

	нулю вещественное число, которое после прибавления его к 1,0 еще дает результат, отличный от 1,0. Тестировать PASCAL следует уменьшением произвольного начального числа на каждом шаге в два раза.
3.	Используя оператор цикла с параметром, постройте орнамент. **OOOOOO**
	Вариант 16
1.	Дано действительное число a, натуральное число n. Вычислить, используя оператор цикла с постусловием, a^n.
2.	Для каждого посетителя парикмахерской (с одним мастером) известны следующие величины: t – момент его прихода и τ – продолжительность его обслуживания. Сколько клиентов обслужит мастер за смену продолжительностью T? Сколько рабочего времени он потратит на обслуживание?
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 17
1.	Дано действительное число a, натуральное число n. Вычислить: $a(a + 1)(a + 2) \ldots (a + n – 1)$.
2.	Проверить численно первый замечательный предел $\lim\limits_{x\to 0}\frac{\sin x}{x}=1$, задавая x значения $1;\frac{1}{2};\frac{1}{4};\frac{1}{8};\ldots$ до тех пор, пока левая часть равенства не будет отличаться от правой менее чем на заданную погрешность ε.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 18
1.	Дано действительное число a, натуральное число n. Вычислить: $\frac{1}{a}+\frac{1}{a(a+1)}+\ldots+\frac{1}{a(a+1)\ldots(a+n)}$.
2.	Проверить численно второй замечательный предел: $\lim\limits_{n\to\infty}\left(1+\frac{1}{n}\right)^n=e$, задавая n значения 1, 2, 3… При каком n исследуемое выражение отличается от e менее чем на заданную погрешность ε?
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 19

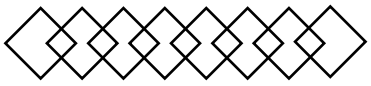
1.	Дано действительное число *a*, натуральное число *n*. Вычислить: $\frac{1}{a}+\frac{1}{a^2}+\frac{1}{a^4}+\ldots+\frac{1}{a^{2n}}$.
2.	Дано действительное число *x*. Вычислить $x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-\frac{x^{11}}{11!}+\frac{x^{13}}{13!}$
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 20
1.	Дано действительное число *a*, натуральное число *n*. Вычислить: $a(a-n)(a-2n)\ \ldots\ (a-n^2)$.
2.	Даны действительные числа *x*, *a*, натуральное число *n*. Вычислить $\underbrace{(\ldots(}_{n\text{ скобок}}(x+a)^2+a)^2+\ldots+a)^2+a)^2+a$
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 21
1.	Вычислить $(1+\sin 0{,}1)(1+\sin 0{,}2)\ldots(1+\sin 10)$.
2.	Дано натуральное число *n*. Вычислить: $\sum_{k=0}^{n}\frac{(-1)^k(k+1)}{k!}$.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 22
1.	Дано действительное число *a*. Найдите среди чисел $1{,}1+\frac{1}{2},1+\frac{1}{2}+\frac{1}{3},\ldots$ первое, большее *a*.
2.	Дано натуральное число *n*. Вычислить: $\sum_{k=1}^{n}\frac{k!}{\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{k+1}}$.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 23
1.	Дано действительное число *a*. Найдите такое наименьшее *n*, что $1+\frac{1}{2}+\ldots+\frac{1}{n}>a$.

2.	Дано натуральное число n, действительное число x. Вычислить: $\prod_{k=1}^{n}\left(\frac{k}{k+1}-\cos^k\lvert x\rvert\right)$.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 24
1.	Дано натуральное n, действительное число x. Вычислить: $\sin x+\sin^2 x+\ldots+\sin^n x$.
2.	Дано натуральное число n, действительное число x. Вычислить: $\prod_{k=1}^{n}\frac{(1-x)^{k+1}+1}{((k-1)!+1)^2}$.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 25
1.	Дано натуральное n, действительное число x. Вычислить $\sin x + \sin x^2 + \ldots + \sin x^n$;
2.	Вычислить бесконечную сумму $\sum_{i=1}^{\infty}\frac{1}{i^2}$ с заданной точностью ε (ε > 0). Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 26
1.	Дано целое число $m > 1$. Получить наибольшее целое k, при котором $4^k < m$.
2.	Вычислить бесконечную сумму $\sum_{i=1}^{\infty}\frac{1}{i\cdot(i+1)}$ с заданной точностью ε ($\varepsilon > 0$). Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 27
1.	Дано натуральное число n. Получить наименьшее число вида 2^r, превосходящее n.
2.	Вычислить бесконечную сумму $\sum_{i=1}^{\infty}\frac{(-1)^i}{i!}$ с заданной точностью ε ($\varepsilon > 0$). Счи-

	тать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 28
1.	Вычислить $\sum_{i=1}^{100} \frac{1}{i^2}$.
2.	Вычислить бесконечную сумму $\sum_{i=1}^{\infty} \frac{(-2)^i}{i!}$ с заданной точностью ε ($\varepsilon > 0$). Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 29
1.	Вычислить $\sum_{i=1}^{50} \frac{1}{i^3}$.
2.	Вычислить бесконечную сумму $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i \cdot (i+1)(i+2)}$ с заданной точностью ε ($\varepsilon > 0$). Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε.
3.	Используя оператор цикла с параметром, постройте орнамент. 
	Вариант 30
1.	Вычислить $\sum_{i=1}^{128} \frac{1}{(2i)^2}$.
2.	Вычислить бесконечную сумму $\sum_{i=1}^{\infty} \frac{1}{4^i + 5^{i+2}}$ с заданной точностью ε ($\varepsilon > 0$). Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε.
3.	Используя оператор цикла с параметром, постройте орнамент. 

Решение типового варианта

Пусть вариант студента 31.

	Вариант 31
1.	Дано действительное число $a > 0$. Последовательность x_0, x_1, ... образована по закону $x_0 = 1$, $x_n = \frac{4}{5}x_{n-1} + \frac{a}{5x_{n-1}^4}$, $n = 1, 2, ...$ Найдите первый член x_n, для которого $\frac{5}{4}a \cdot \lvert x_{n+1} - x_n \rvert < 10^{-6}$. Вычислите для найденного значения x_n разность $a - x^5_n$.
2.	Дано натуральное число n, действительное число x. Вычислите: $\sum_{i=1}^{n} \frac{x + \cos(i \cdot x)}{2^i}$.
3.	Используя оператор цикла с параметром, постройте орнамент. 

Сформируем отчет к лабораторной работе 4.

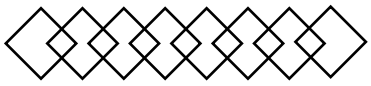
1. Титульный лист.

2. Задание варианта 31.

1. Дано действительное число $a > 0$. Последовательность x_0, x_1, ... образована по закону $x_0 = 1$, $x_n = \frac{4}{5}x_{n-1} + \frac{a}{5x_{n-1}^4}$, $n = 1, 2, \ldots$ Найдите первый член x_n, для которого $\frac{5}{4}a \cdot \lvert x_{n+1} - x_n \rvert < 10^{-6}$. Вычислите для найденного значения x_n разность $a - x^5_n$.

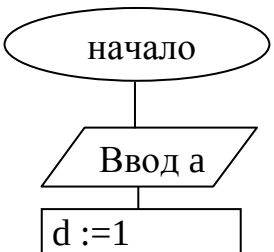
2. Дано натуральное число n, действительное число x. Вычислите: $\sum_{i=1}^{n} \frac{x + \cos(i \cdot x)}{2^i}$

3. Используя оператор цикла с параметром, постройте орнамент.

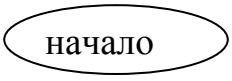


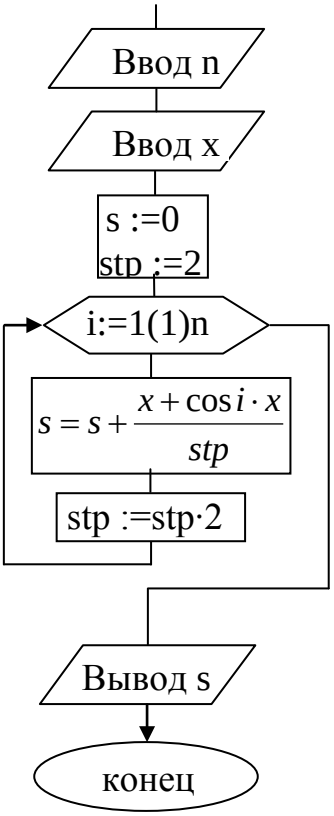
4. Составим блок-схему алгоритмов заданий варианта.

1.



2.





5. Напишем программы.

```
{1.} Uses crt;
  Var
    a, t, d: real;
    n: integer;
  Begin
    ClrScr;
    Write ('Введите a:');
    Readln (a);
    d := 1;
    t:=4/5 * d + a / 5 / sqr (sqr(d));
    n := 1;
```

```
    While Abs (t-d)>1e-6 do begin
      d := t;
      t:= 4/5 * d + a / 5 / sqr (sqr(d));
      n := n + 1;
    End;
    Write ('a – t5=', a – t*t*t*t*t: 8: 7);
    Readkey;
  End.
{2.} Uses crt;
    Var
      i, n: integer;
      s, x, stp: real;
    Begin
      Write ('Введите n:');
      Readln (n);
      Write ('Введите x:');
      Readln (x);
      S:=0;
      stp:= 2;
      For i:=1 to n do begin
        S := s + (x + cos (i * x))/stp;
        stp := stp * 2
      End;
      Write ('s=', s:6:3);
      Readkey;
    End.
{3.} Uses crt, graph;
     Var
       Gd, Gm, i, s: Integer;
     Begin
       Gd := detect;
       InitGraph (Gd, Gm, '');
       S:= 60;
```

```
        For i :=0 to 7 do begin
         Line (50 + i * s, 100, 100 + i * s, 50);
         Line (150 + i * s, 100, 100 + i * s, 50);
         Line (100 + i * s, 150, 150 + i * s, 100);
         Line (100 + i * s, 150, 50 + i * s, 100);
        end;
        readkey;
        CloseGraph;
end.
```

6. Представленные программы не содержат ошибок, подберем значения для тестирования программ.

1) Если A = 3, тогда Xn = 1.245731;

Если A = 50, тогда Xn = 2.186724.

2) Если N = 3, X = 2, тогда S = 1.499;

Если N = 50, X = 5, тогда S = 4.888.

3) Задание с графикой тестируется визуально.

7. В текстовом редакторе WORD оформлен отчет в соответствии с содержанием.

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1. Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Радер, Дж. Бейсик для персонального компьютера фирмы IBM / Дж. Радер, К. Миллсап ; пер. с англ. – М. : Радио и связь, 1991. – 30 л.: ил.

3. Абрамов, С. А. Начала информатики / С. А. Абрамов, Е. В. Зима. – М. : Наука, 1989. – 256 с.

4. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

5. Довгаль, С. И. Персональные ЭВМ : Турбо Паскаль V 7.0. Объектное программирование. Локальные сети : учебное пособие / С. И. Довгаль, Б. Ю. Литвинов, А. И. Сбитнев.

6. Епанешников, А. Программирование в среде Turbo Pascal 7.0 / А. Епанешников, В. Епанешников. – М. : «ДИАЛОГ-МИФИ», 1993. – 288 с.

7. Новичков, В. С. Паскаль : учеб. пособие для сред. спец. учеб. заведений / В. С. Новичков, Н. И. Парфилова, А. Н. Пылькин. – М. : Высш. шк., 1990. – 223 с. : ил. – (Алгоритмические языки в технику-ме).

8. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

9. Хершель, Рудольф. Турбо Паскаль / Рудольф Хершель. – 2-е изд., перераб. – Вологда : МП «МИК», 1991. – 342 с. при участии МП ТПО «Квадрат», г. Москва.

13. Эрбс, Х.-Э. Введение в программирование на языке Паскаль / Х.-Э. Эрбс, О. Штольц ; пер. с нем. – М. : Мир, 1989. – 299 с., ил.

14. Григас, Г. Начала программирования : кн. для учащихся ; пер. с лит. / Г. Григас ; под. ред. Ю. А. Первина.– М. : Просвещение, 1987. – 112 с. : ил.

15. Могилев, А. В. Информатика : учеб. пособие для студ. пед. вузов / А. В Могилев, Н. И. Пак, Е. К. Хеннер ; под ред. Е. К. Хеннера. – М., 1999. – 816 с.

Вопросы для самопроверки

1. Дайте понятие цикла с предусловием. Укажите служебные слова, его оформляющие, и приведите пример оператора, в котором он используется. Приведите его графическое изображение.

2. Дайте понятие цикла с постусловием. Укажите служебные слова, его оформляющие, и приведите пример оператора, в котором он используется. Приведите его графическое изображение.

3. Дайте понятие цикла с заданным числом повторений. Укажите служебные слова, его оформляющие, и приведите пример операто-

ра, в котором он используется. Приведите его графическое изображение.

4. Напишите фрагмент программы вывода на экран целых чисел от 1 до 10 с использованием оператора цикла с заданным числом повторений.

5. Напишите фрагмент программы вывода на экран целых чисел от 1 до 10 с использованием оператора цикла с предусловием.

6. Напишите фрагмент программы вывода на экран целых чисел от 1 до 10 с использованием оператора цикла с постусловием.

7. Напишите фрагмент программы с использованием оператора цикла, который выводит на экран числа от 1 до 3 с шагом 0,3 (то есть: 1; 1,3; 1,6 и так далее до 3).

8. Напишите последовательность операторов, способную выполнить операцию задержки выполнения программы до нажатия на любую клавишу.

9. Дан фрагмент программы: **FOR I := 1 TO 10 DO; WRITE(I) ;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

10. Дан фрагмент программы: **FOR I := 10 TO 0 DO WRITE (I) ;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

11. Дан фрагмент программы: **FOR I := 'A' TO 'C' DO WRITE (I:2) ;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

12. Дан фрагмент программы: **A := 5; WHILE A<3 DO A := A – 1; WRITE (A);** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

13. Дан фрагмент программы: **A := 5; WHILE A>3 DO; A := A – 1; WRITE (A);** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

14. Дан фрагмент программы: **A := 5; WHILE NOT (A<3) DO A := A – 1; WRITE (A);** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

15. Дан фрагмент программы: **A := 5; WHILE A>3 DO BEGIN A :=A – 1; WRITE (A) END;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

16. Дан фрагмент программы: **A := 5; REPEAT A :=A – 1 WRITE (A) UNTIL A<3;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

17. Дан фрагмент программы: **A := 5; REPEAT A :=A + 1; WRITE (A) UNTIL A>3;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

18. Дан фрагмент программы: **A := 5; REPEAT A :=A – 1 UNTIL A<3; WRITE (A);** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

19. Дан фрагмент программы: **A := 5; REPEAT WRITE (A) UNTIL A>3;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

20. Дан фрагмент программы: **A:=5; REPEAT WRITE (A) UNTIL A<3;** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

ЛАБОРАТОРНАЯ РАБОТА 5.
ОПЕРАТОР УСЛОВНОГО ПЕРЕХОДА (2 ЧАСА)

Цель работы: освоить простые ветвления в программе. Приобрести навыки работы со структурой управляющих конструкций алгоритмических языков программирования – условным переходом. Изучить конструкции выбора: «если… то…» if … then …, «если … то …

иначе …» if … then … else … и оператор варианта «выбор … из … « case … of ….

Теоретические положения

Полная форма условного оператора

Часто бывает необходимо в зависимости от ситуации, возникшей в ходе решения задачи, выбрать один вариант решения из двух или более возможных.

Пример.

Написать программу вычисления по формуле $y = \sqrt{x}$.

Решением задачи может служить приведенная ниже программа.

```
Var
x: real;
BEGIN
  Writeln ('Введи x ');
  Readln (x);
  Writeln (‘sqrt (‘, x, ’) = ', sqrt (x));
END.
```

Если ввести с клавиатуры неотрицательное значение, то будет вычислено значение функции, иначе система выдаст сообщение об ошибке и произойдет аварийный останов программы. Естественней было бы не прерывать программу, а выдать сообщение типа «Функция не определена» и закончить работу естественным путем.

Подобные ситуации встречаются в программировании очень часто. Необходимо бывает проверить некоторое условие, наложенное на переменную. Если же значение переменной условию не удовлетворяет, выполняется другая группа операторов. Для реализации таких ситуаций используется условный оператор

IF B **THEN** P1 **ELSE** P2,

где B – логическое условие, P1, P2 – простые или составные операторы;

THEN, ELSE – служебные слова (то, иначе), после которых может выполниться один простой или составной оператор.

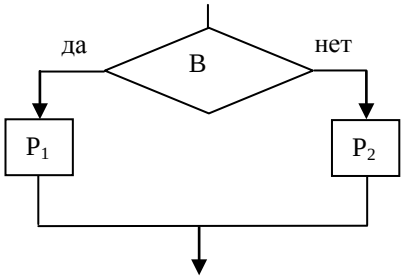


Рис. 1. На языке блок-схем

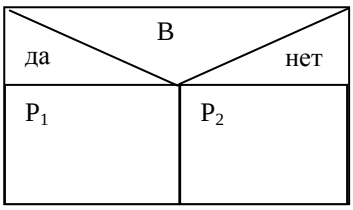


Рис. 2. В структурограммах по Насси и Шнейдерману

Таким образом, с использованием оператора выбора программа примет следующий вид:

```
Var
  x: real;
BEGIN
   Write ('Введи x:');
   Readln (x);
   IF x Б < > 0 THEN writeln (‘sqrt (‘, x, ’)=', sqrt (x))
            ELSE writeln('Функция не определена в точке x=’, x);
END.
```

Сокращенный условный оператор

В общем случае условный оператор состоит из условия, расположенного вслед за служебным словом IF, и двух операторов, из которых только один выполняется в зависимости от истинности условия. Иногда оказывается полезным упрощенный вариант условного оператора: необходимо выполнить некоторое действие только при истинности проверяемого условия. В таком случае один из операторов можно опустить.

То есть допускается в программах использовать сокращенную форму условного оператора

IF B **THEN** P**.**

где B – логическое условие, S – простой или составной оператор;

THEN – служебное слово (то), после которого может выполниться один простой или составной оператор.

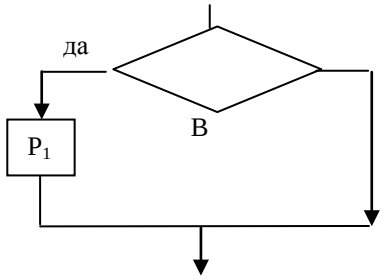


Рис. 3. На языке блок-схем

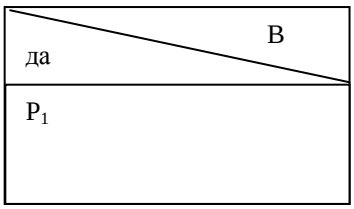


Рис. 4. В структурограммах по Насси и Шнейдерману

Пример.

Дано действительное значение *x*. Вычислить значение функции

$$f(x)=\begin{cases} x^2+x-2, \text{если } x \ge 0; \\ \dfrac{x+1}{x^2+x-2}, \text{если } x < 0. \end{cases}$$

```
Var
  x, y: real;
BEGIN
   Write ('Введи x:');
   Readln (x);
   y := sqr(x) + x – 2;
   IF x < 0 THEN y := (x + 1)/y;
   writeln (‘f(x)=', y)
END.
```

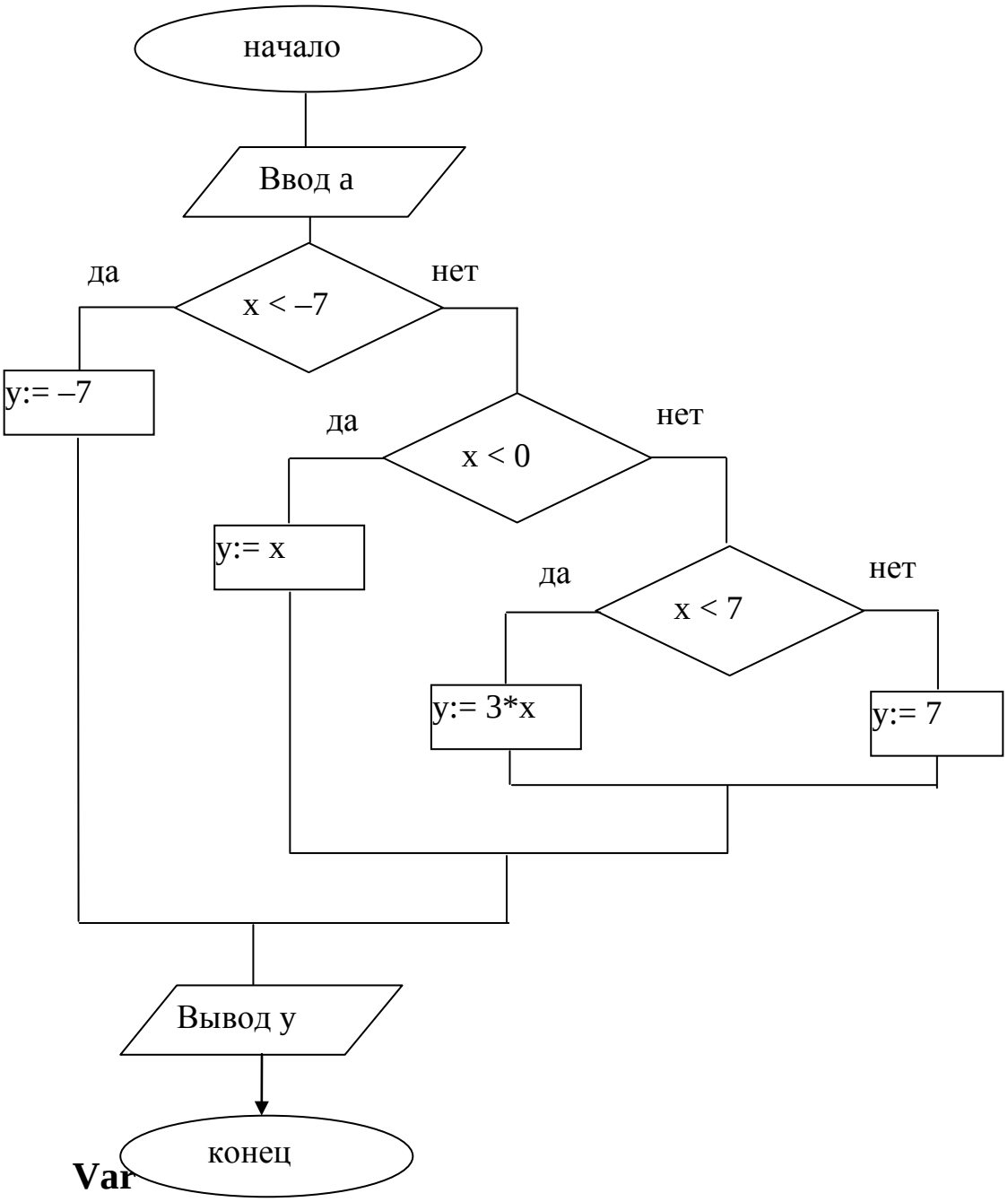
Многозначные ветвления в программах

Часто приходится выбирать путь решения задачи не из двух, а из нескольких возможных. В программировании это можно реализовать, используя несколько условных операторов. В этом случае после служебных слов THEN и ELSE записывается новый условный оператор. Так можно писать, поскольку условный оператор равноправен с другими операторами.

Пример.

Зависимость результата *y* от исходного действительного *x* выражается следующей зависимостью $y=\begin{cases} -7, \text{если } x < -7; \\ x, \text{если } -7 \le x < 0; \\ 3x, \text{если } 0 \le x < 7; \\ 7, \text{если } x \ge 7. \end{cases}$. Написать программу, вычисляющую значение функции *y* от *x*.

Разбить числовую прямую на четыре интервала и выделить нужный можно разными способами. Самый очевидный – пройти условия сверху вниз, как они записаны.




```
Var
  x, y: real;
BEGIN
  Write ('Введи x:');
  Readln (x);
  IF x < –7 THEN y := –7
          ELSE IF x < 0 THEN y := x
                        ELSE IF x < 7 THEN y := 3* x
```

```
                                    ELSE y := 7;
    writeln('y=', y);
END.
```

Оператор **IF** реализует алгоритмическую конструкцию РАЗВИЛКА (или условный переход) и изменяет порядок выполнения операторов в зависимости от истинности или ложности некоторого условия.

Условие обычно записывается в виде равенства или неравенства.

Логические операции

Логические операция применяются к величинам логического типа, результат операции тоже логического типа. Имеется одна унарная логическая операция **NOT** (ОТРИЦАНИЕ) и три бинарные операции **AND** (И), **OR** (ИЛИ), **XOR** (ИСКЛЮЧАЮЩЕЕ ИЛИ). Они определяются следующими таблицами истинности (табл. 1-4).

Таблица 1

NOT	
x	***not** x*
True	*false*
False	*true*

Таблица 2

AND		
x	*y*	*x **and** y*
true	*true*	*true*
true	*false*	*false*
false	*true*	*false*
false	*false*	*false*

Таблица 3

OR		
x	*y*	*x **or** y*
True	*true*	*true*
True	*false*	*true*
False	*true*	*true*
False	*false*	*false*

Таблица 4

XOR		
x	*y*	*x **xor** y*
true	*true*	*false*
true	*false*	*true*
false	*true*	*true*
false	*false*	*false*

Излюбленный вариант начинающих – полная проверка всех условий – тоже может использоваться, но только с использованием логических связок. Нужно помнить, что каждое условие в связке должно быть в скобках. Таким образом, есть вариант решения последнего примера

```
Var
  x, y: real;
BEGIN
   Writeln ('Введи x');
   Readln (x);
   IF x < –7 THEN y := –7;
   IF (–7 <= x) AND (x < 0) THEN y := x;
   IF (0 <= x) AND (x < 7)  THEN y := 3 * x;
   IF x > 7 THEN y := 7;
      writeln('y=’, y);
END.
```

Этот вариант всего лишь допустим, но его следует избегать, так как такое программирование может порождать неочевидную логическую ошибку при неполном или ошибочном переборе всех случаев. Основная рекомендация большинства методических изданий – всегда использовать полную версию оператора условного перехода.

Оператор варианта

С помощью оператора варианта **CASE** можно выбрать вариант из любого количества вариантов. Структура этого оператора в Turbo Pascal:

```
CASE Select OF
   cl: P1;
   c2: P2;
```

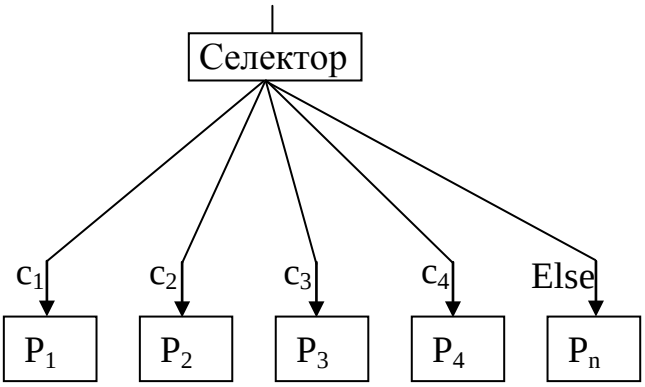
.....

cn: Pn;

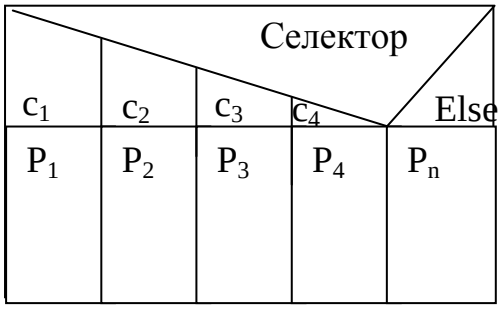
ELSE S

END;

На языке блок-схем:

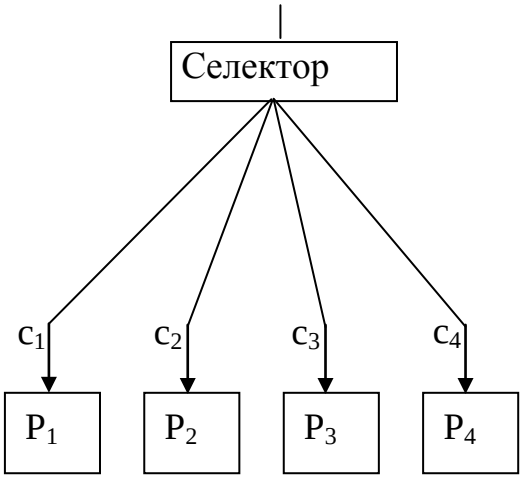


В структурограммах по Насси и Шнейдерману:

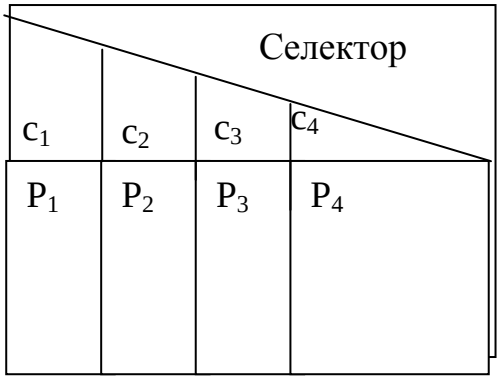


В этой структуре ветвь оператора **ELSE** является необязательной. Если она отсутствует и значение выражения Select не совпадет ни с одной из перечисленных констант, весь оператор рассматривается как пустой.

На языке блок-схем:



В структурограммах по Насси и Шнейдерману:



Порядок выполнения работы

1. Уточните номер своего варианта.
2. Составьте блок-схему алгоритмов заданий варианта.
3. Напишите программу по составленной блок-схеме.
4. Отладьте программу (исключите все сообщения об ошибках) и подберите значения для тестирования программ (запишите результаты тестирования).

5. В текстовом редакторе WORD или рукописно оформите отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.
2. Задания варианта
3. Блок-схема алгоритмов заданий варианта.
4. Программа по составленной блок-схеме.
5. Результат выполнения программы.
6. Описание использовавшихся операций копирования строк.

Варианты задач

	Вариант № 1
1.	Даны действительные числа *x*, *y*. Вычислите значение *z*: $z = \begin{cases} x - y, & x > y \\ y - x + 1, & x \le y \end{cases}$
2.	Дано натуральное число *n*, задан входной поток данных: целые числа a_1, a_2, a_3, …, a_n. Получите сумму положительных и число отрицательных членов входного потока данных.
	Вариант 2
1.	Даны действительные числа *x*, *y*, *z*. Вычислите значение $\min^2(x + y + 0{,}5 \cdot z; xyz) + 1$.
2.	Дано натуральное число *n*, , задан входной поток данных: действительные числа a_1, a_2, a_3, …, a_n. Просуммируйте все неотрицательные члены, принадлежащие отрезку [1, 2]. Если таких чисел нет, выведите на экран единицу.
	Вариант 3
1.	Даны действительные числа *a, b, c*. Проверьте, выполняются ли неравенства $a < b < c$.
2.	Дано натуральное число *n*, задан входной поток данных: действительные числа x_1, x_2, x_3, …, x_n. Во входном потоке данных x_1, x_2, x_3, …, x_n, получите сумму членов, принадлежащих отрезку [3, 7], а также число таких членов. Выдайте на экран сообщение, есть ли в потоке хотя бы один ноль.
	Вариант 4
1.	Даны действительные числа *a, b, c*. Увеличьте их в два раза, если $a \ge b \ge c$, и замените их квадратами, если это не так.
2.	Дано натуральное число *n*, задан входной поток данных: действительные числа a_1, a_2, a_3, …, a_n. Во входном потоке данных a_1, a_2, a_3, …, a_n найдите количество отрицательных членов и сумму всех неотрицательных.
	Вариант 5
1.	Даны два действительных числа. Выведите на экран первое число, если оно больше второго, и оба числа, если это не так.
2.	Дано натуральное число *n*, входной поток данных a_1, a_2, a_3, …, a_n. Вычислите обратную величину произведения трех первых членов a_i последовательности a_1, a_2, a_3, …, a_n, для которых выполнено $i < a_i < i^2$.
	Вариант 6
1.	Даны два действительных числа. Замените первое число нулем, если оно меньше или равно второму, и оставьте числа без изменения в противном случае.
2.	Дано натуральное число *n*, входной поток данных: действительные числа a_1, a_2, a_3, …, a_n. Получите удвоенную сумму всех положительных членов входного потока данных a_1, a_2, a_3, …, a_n.
	Вариант 7
1.	Даны три действительных числа. Выберите из них те, которые принадлежат интервалу (1; 3).
2.	Даны целые числа *p*, *q*, входной поток данных: целые числа a_1, a_2, a_3, …, a_{64}

	($p > q \geq 0$). Во входном потоке данных a_1, a_2, a_3, …, a_{64} вычислите количество членов, модуль которых при делении на p дает в остатке q. При вводе такого числа выдайте на экран соответствующее сообщение.
	Вариант 8
1.	Даны действительные числа *x, y* ($x \neq y$). Меньшее из этих двух чисел замените их полусуммой, а большее – их удвоенным произведением.
2.	Даны натуральные числа n, p, входной поток данных: целые числа a_1, a_2, a_3, …, a_n. Получите произведение членов входного потока данных a_1, a_2, a_3, …, a_n кратных p.
	Вариант 9
1.	Даны три действительные числа. Возведите в квадрат те из них, значения которых неотрицательны.
2.	Дано натуральное число n, входной поток данных: целые числа a_1, a_2, a_3, …, a_n . Найдите количество и сумму тех членов данного входного потока данных, которые делятся на 5 и не делятся на 7.
	Вариант 10
1.	Если сумма трех попарно различных действительных чисел *x, y, z* меньше единицы, то наименьшее из этих трех чисел замените полусуммой двух других; в противном случае замените меньшее из *x* и *y* полусуммой двух оставшихся значений.
2.	Дан входной поток данных: целые числа числа a_1, a_2, a_3, …, a_{50}. Получите сумму трех первых чисел данного входного потока данных, которые удовлетворяют условию $\mid a_i \mid < i^2$.
	Вариант 11
1.	Даны действительные числа *a, b, c, d*. Если $a \leq b \leq c \leq d$, то каждое число замените наибольшим из них; если $a > b > c > d$, то числа оставьте без изменения; в противном случае все числа замените их квадратами.
2.	Дан входной поток данных: целые числа a_1, a_2, a_3, …, a_{50}. Получите сумму трех первых чисел данного потока данных, которые нечетны и отрицательны.
	Вариант 12
1.	Даны действительные положительные числа *x, y, z*. Выяснить, существует ли треугольник с длинами сторон *x, y, z*, и если треугольник существует, то ответить является ли он остроугольным.
2.	Даны действительные числа *x, y*. Если *x* и *y* отрицательны, то каждое значение замените его модулем; если отрицательно только одно из них, то оба значения увеличьте на 0,5; если оба значения неотрицательны и ни одно из них не принадлежит отрезку [0,5; 2,0], то оба значения уменьшьте в 10 раз; в остальных случаях *x* и *y* оставьте без изменения.
	Вариант 13
1.	Даны действительные числа *a, b, c* ($a \neq 0$). Выясните, имеет ли уравнение $ax^2 + bx + c = 0$ действительные корни. Если действительные корни имеются, то найдите их. В противном случае ответом должно служить сообщение, что действительных корней нет.
2.	Дан входной поток данных: целые числа a_1, a_2, a_3, …, a_{50}. Получите сумму трех первых чисел данной последовательности, которые кратны 5.
	Вариант 14
1.	Дан входной поток данных: действительные числа a_1, b_1, c_1, a_2, b_2, c_2. Выясните, верно ли, что $\lvert a_1b_2 - a_2b_1 \rvert \geq 0.0001$, и если верно, то найдите решение системы уравнений $\begin{cases} a_1x^2 + b_1x + c_1 = 0 \\ a_2x^2 + b_2x + c_2 = 0 \end{cases}$.
2.	Даны действительные положительные числа *a, b, c, x, y*. Выясните, пройдет ли

	кирпич с ребрами *a, b, c* в прямоугольное отверстие со сторонами *x* и *y*. Просовывать кирпич в отверстие разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.
	Вариант 15
1.	Дано действительное число *a*. Вычислите *f*(*a*), если $f(x)=\begin{cases} x^2, \text{при} -2 \le x < 2; \\ 4, \text{в противном случае.} \end{cases}$
2.	Дано натуральное число n . Получите сумму трех чисел вида $i^3 - 3in^2 + n$ (i = 1, 2,… , n), которые являются утроенными нечетными.
	Вариант 16
1.	Дано действительное число *a*. Вычислите *f*(*a*), если $f(x)=\begin{cases} 0, \text{при } x \le 0; \\ x, \text{при } 0 < x \le 1; \\ x^4, \text{в остальных случаях} \end{cases}$
2.	Дано натуральное число *n*, задан входной поток данных: вещественные числа $q_1, q_2, q_3, \ldots, q_n$. Найдите те члены q_i входного потока данных, у которых корни уравнения $x^2 + 3q_i - 5 = 0$ действительны и положительны.
	Вариант 17
1.	Дано действительное число *a*. Вычислите *f*(*a*), если $f(x)=\begin{cases} x^2+4x+5, \text{при } x \le 2; \\ \dfrac{1}{x^2+4x+5}, \text{в противном случае.} \end{cases}$
2.	Дано натуральное число *n*, задан входной поток данных: натуральные числа $q_1, q_2, q_3, \ldots, q_n$. Найдите те члены q_i входного потока данных, которые при делении на 7 дают остаток 1, 2 или 5.
	Вариант 18
1.	Дано действительное число *a*. Вычислить *f*(*a*), если $f(x)=\begin{cases} 0, & \text{при } x \le 0; \\ x^2 - x, & \text{при } 0 < x \le 1; \\ x^2 - \sin \pi x^2, & \text{в остальных случаях.} \end{cases}$
2.	Дано натуральное число *n*, задан входной поток данных: натуральные числа $q_1, q_2, q_3, \ldots, q_n$. Найдите те члены q_i входного потока данных, которые являются удвоенными нечетными числами.
	Вариант 19
1.	Даны действительные числа *x, y*. Получите max(min(*x*; *y*) + 5; 0,3·max(*x*; *y*)).
2.	Дано натуральное число *n*, задан входной поток данных: натуральные числа $a_1, a_2, a_3, \ldots, a_n$. Определите количество членов a_k входного потока данных, имеющих четные порядковые номера и являющихся нечетными числами.
	Вариант 20
1.	Даны действительные числа *x, y*. Получите min(6 · min(2 · *x*; *y*); *x*) · max(*x*; *y*)).
2.	Дано натуральное число *n*, задан входной поток данных: натуральные числа $a_1, a_2, a_3, \ldots, a_n$. Определите количество членов a_k входного потока данных, удовлетворяющих условию $2^k < a_k < k$!
	Вариант 21
1.	Дано действительное число *a*. Вычислите *f*(*a*), если

	$f(x)=\begin{cases} x^2, & \text{при } x \le -1; \\ x, & \text{при } -1 < x \le 3; \\ \operatorname{arctg}\sqrt{x}, & \text{в остальных случаях.} \end{cases}$
2.	Дано натуральное число n, задан входной поток данных: натуральные числа a_1, a_2, a_3, …, a_n. Определите количество членов a_k входного потока данных, удовлетворяющих условию $a_k < \frac{a_{k-1} + a_{k+1}}{2}$.
	Вариант 22
1.	Дано действительное число a. Вычислите $f(a)$, если $f(x)=\begin{cases} \sqrt[3]{x}, & \text{при } x \le 1; \\ \sqrt[3]{x^2}, & \text{при } 1 < x \le 2; \\ \sqrt[3]{\sin^2 \pi}, & \text{в остальных случаях.} \end{cases}$
2.	Дано натуральное число n, задан входной поток данных: натуральные числа a_1, a_2, a_3, …, a_n. Определите количество членов a_k входного потока данных являющихся квадратами четных чисел.
	Вариант 23
1.	Дано действительное число a. Вычислите $f(a)$, если $f(x)=\begin{cases} \cos x, & \text{при } -\pi < x \le 0; \\ \sin x^2, & \text{при } 0 < x \le \pi \\ \operatorname{tg}^2 x^2, & \text{в остальных случаях.} \end{cases}$
2.	Дано натуральное число n, задан входной поток данных: натуральные числа a_1, a_2, a_3, …, a_n. Определите количество членов a_k входного потока данных кратных 3 и не кратных 5.
	Вариант 24
1.	Дано действительное число a. Вычислите $f(a)$, если $f(x)=\begin{cases} \cos\lvert x\rvert, & \text{при } -1{,}57 < x \le 0; \\ \sin(-x), & \text{при } 0 < x \le 1{,}57; \\ \operatorname{arctg} x^2, & \text{в остальных случаях.} \end{cases}$
2.	Дано натуральное число n, задан входной поток данных: натуральные числа a_1, a_2, a_3, …, a_n. Определите количество членов a_k входного потока данных, являющихся нечетными числами.
	Вариант 25
1.	Дано действительное число a. Вычислите $f(a)$, если $f(x)=\begin{cases} \lvert x\rvert, & \text{при } x \le 0; \\ x, & \text{при } 0 < x \le 1; \\ \{x\}, & \text{в остальных случаях.} \end{cases}$ $\lvert x\rvert$ – модуль числа x, $\{x\}$ – дробная часть числа x.
2.	Даны действительные положительные числа a, b, c, d. Выясните, можно ли прямоугольник со сторонами a, b уместить внутри прямоугольника со сторонами c, d так, чтобы каждая из сторон одного прямоугольника была параллельна или перпендикулярна каждой стороне второго прямоугольника.
	Вариант 26
1.	Даны действительные числа x, y. Получите $\min(\max(x^2; y);20/\min(x;y))$.

2.	Даны действительные числа x_1, x_2, x_3, y_1, y_2, y_3. Определите, принадлежит ли начало координат треугольнику с вершинами (x_1, y_1), (x_2, y_2), (x_3, y_3)?
	Вариант 27
1.	Даны действительные числа *x, y*. Получите $\sqrt{\max\left(x^2, \sqrt{y}\right) - \min\left(\sqrt{x}, y^2\right)}$.
2.	Даны действительные числа *a, b, c, d, e, f, g, h*. Известно, что точки (*e, f*) и (*g, h*) различны. Известно также, что точки (*a, b*) и (*c, d*) не лежат на прямой *l*, проходящей через точки (*e, f*) и (*g, h*). Прямая *l* разбивает координатную плоскость на две полуплоскости. Выясните, верно ли, что точки (*a, b*) и (*c, d*) принадлежат одной и той же полуплоскости.
	Вариант 28
1.	Даны вещественные числа *x, y*. Получите min(min(*x*; *y*) + 5; *x* · 5 – min(*x*; *y*)).
2.	Даны действительные числа *a, b, c, d, s, t, u* (*s* и *t* одновременно не равны нулю). Известно, что точки (*a, b*) и (*c, d*) не лежат на прямой *l*, заданной уравнением $sx + ty + u = 0$. Прямая *l* разбивает координатную плоскость на две полуплоскости. Выясните, верно ли, что точки (*a, b*) и (*c, d*) принадлежат разным полуплоскостям.
	Вариант 29
1.	Даны действительные числа *x, y*. Получите $\max^3(\min(x + 2; y - 3); \max(x; y))$.
2.	Дано действительное число *h*. Выясните, имеет ли уравнение $ax^2 + bx + c = 0$ действительные корни, если $a = \sqrt{\frac{\left\|\sin 8h\right\| + 17}{\left(1 - \sin 4h \cdot \cos\left(h^2 + 18\right)\right)^2}}$, $b = 1 - \sqrt{\frac{3}{3 + \left\|\operatorname{tg} ah^2 - \sin ah\right\|}}$, $c = ah^2 \sin bh + bh^3 \cos ah$. Если действительные корни существуют, то найдите их. В противном случае ответом должно служить сообщение, что действительных корней нет.
	Вариант 30
1.	Напишите программу для вычисления значений функции $y = \begin{cases} \frac{\sin x}{x}, \text{если } x > 0 \\ \cos x, \text{если } x \le 0 \end{cases}$
2.	Даны действительные числа *a, b, c* ($a \neq 0$). Полностью исследуйте биквадратное уравнение $ax^4 + bx^2 + c = 0$, то есть если действительных корней нет, то должно быть выдано сообщение об этом, иначе должны быть выданы два или четыре корня.

Решение типового варианта

Рассмотрим пример оформления работы.

Пусть вариант студента 31.

	Вариант 31
1.	Даны действительные числа *x, y, z*. Вычислите max(*x, y, z*) – min(*x, y, z*).
2.	Дано натуральное число *n*, задан входной поток данных: действительные числа a_1, a_2, a_3, …, a_n. Просуммируйте все неотрицательные члены, большие единицы. Если таких чисел нет, выведите на экран последовательность символов 000.

Сформируем отчет к лабораторной работе 5.

1. Титульный лист.

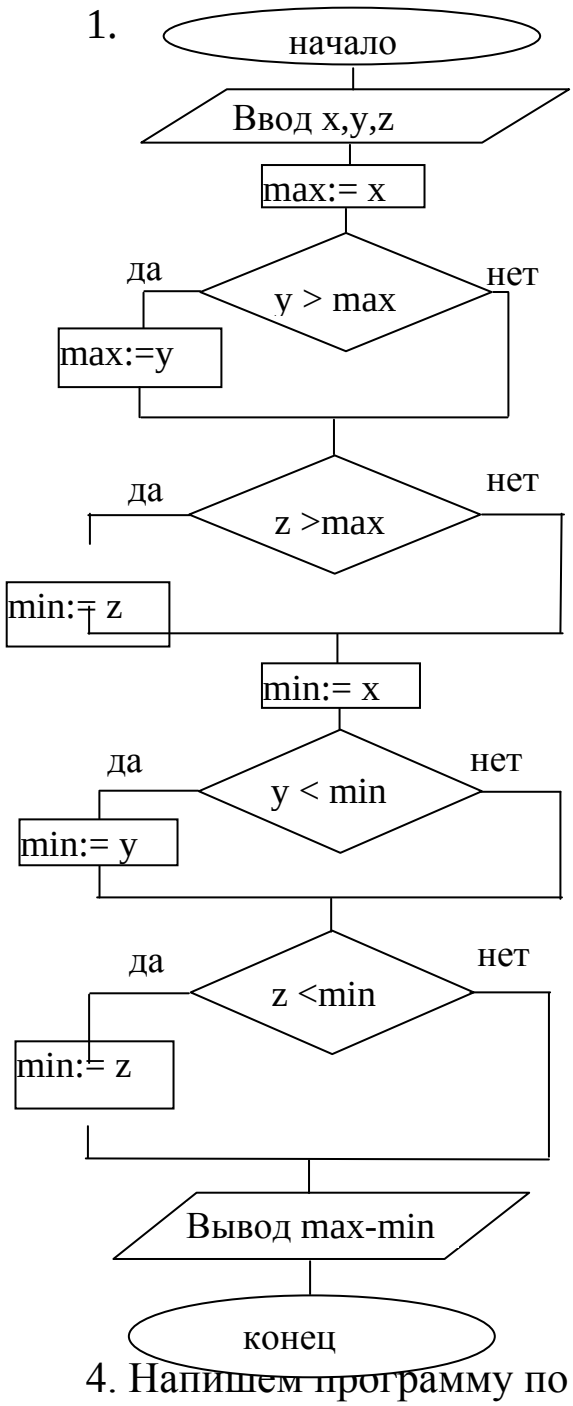
2. Сформулируем задание варианта 31.

1. Даны действительные числа *x*, *y*, *z*. Вычислите max(*x*, *y*, *z*) – min(*x*, *y*, *z*);

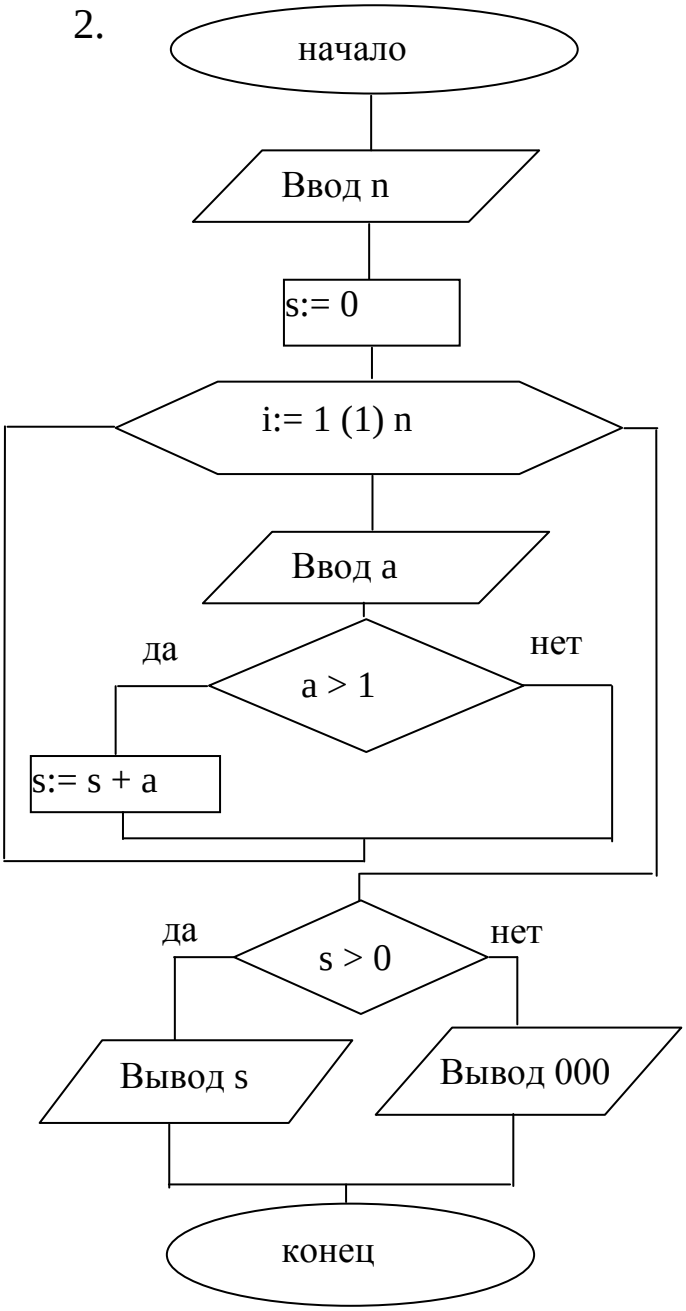
2. Дано натуральное число *n*, задан входной поток данных: действительные числа a_1, a_2, a_3, …, a_n. Просуммируйте все неотрицательные члены, большие единицы. Если таких чисел нет, выведите на экран последовательность символов 000.

3. Составим блок-схему алгоритмов заданий варианта.

1.



2.



4. Напишем программу по составленной блок-схеме.

```
{1.}
```

```
  uses crt;
  var
    x,y,z,max,min:real;
begin
  write('Введите x,y,z:');
  readln(x,y,z);
  max:=x;
  if y>max then max:=y;
  if z>max then max:=z;
  min:=x;
  if y<min then min:=y;
  if z<min then min:=z;
  Writeln('max-min=',max-min:5:3);
  readkey;
end.
{2.} uses crt;
var
i,n:integer;
s,a:real;
begin
  write('Введите n:');
  readln(n);
  s:=0;
  for i:=1 to n do begin
    write('Введите a[',i,']');
    readln(a);
    if a>1 then s:=s+a;
  end;
  if s>0 then writeln('s=',s:6:3)
         else writeln('000');
  readkey;
end.
```

5. Представленные программы не содержат ошибок, подберем значения для тестирования программ.

1. x = 25, y = 76, z = 13 max – min = 63;
   x = –36, y = 5, z = 16 max – min = 52.
2. n = 7, a_i = –3; 5; 0; 2; 1; 0; 3 s = 7
   n = 3, a_i = 36; 16; –72 s = 52
   n = 3, a_i = 0,36; 0,16; –72 000

6. В текстовом редакторе WORD оформим отчет в соответствии с содержанием.

Вопросы для самопроверки

1. Перечислите логические операции на языке Паскаль. Приведите таблицу истинности операции **NOT**.

2. Дайте понятие составного оператора. Укажите служебные слова, его оформляющие, и приведите пример оператора (хотя бы одного) в котором он используется.

3. Дайте понятие условного оператора. Укажите служебные слова, его оформляющие, и приведите пример оператора (хотя бы одного) в котором он используется. Приведите его графическое изображение на блок-схемах.

4. Опишите ситуации, в которых есть необходимость в операторе варианта. Укажите служебные слова, его оформляющие, и приведите пример оператора варианта, в котором в зависимости от значения переменной *a* на экран выводится его символьное изображение (‘ОДИН’, ‘ДВА’, ‘ТРИ’, при любом другом значении слово ‘ДАЛЬШЕ’).

5. Напишите фрагмент программы определения четности числа с использованием оператора условного перехода.

6. Перечислите логические операции на языке Паскаль. Приведите таблицу истинности операции **OR**.

7. Перечислите логические операции на языке Паскаль. Приведите таблицу истинности операции **AND**.

8. Перечислите логические операции на языке Паскаль. Приведите таблицу истинности операции **XOR**.

9. Обоснуйте необходимость для языка программирования операций отношений. Укажите, какими символами какие операции отношений обозначаются.

10. Обоснуйте необходимость ввода понятия пустого оператора в синтаксис Паскаля. Дайте определение пустого оператора в Паскале

11. Дан фрагмент программы: **A:=5; IF NOT (A<3) THEN WRITE(A) ; ELSE WRITE(SQR(A));** Укажите, что появится на экране в результате работы этого фрагмента. Обоснуйте, почему появится это сообщение.

12. Дан фрагмент программы: **A:=5; IF A<3 OR A>10 THEN WRITE(A) ELSE WRITE(SQR(A));** Укажите, что появится на экране в результате работы этого фрагмента. Обоснуйте, почему появится это сообщение.

13. Дан фрагмент программы: **A:=5; IF NOT (A>10) THEN WRITE(A) ELSE WRITE(SQR(A));** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

14. Дан фрагмент программы: **A:=5; IF (A<3) AND (A>10) THEN WRITE(A) ELSE WRITE(SQR(A));** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

15. Дан фрагмент программы: **A:=5; IF NOT(A<3 OR A>10) THEN WRITE(A) ELSE WRITE(SQR(A));** Укажите, что появится на экране в результате работы этого фрагмента. Обоснуйте, почему появится это сообщение.

16. Дан фрагмент программы: **A:=5; IF A>3 THEN WRITE(A) ELSE ; WRITE(SQR(A));** Покажите данный фрагмент на блок-схеме и укажите, что появится на экране в результате работы этого фрагмента.

ЛАБОРАТОРНАЯ РАБОТА 6.
ПОНЯТИЕ СТРУКТУРИРОВАННЫХ ТИПОВ. МАССИВЫ. МАТРИЦЫ. МНОГОМЕРНЫЕ МАССИВЫ (6 ЧАСОВ)

Цель работы: изучить особенности структурированного типа массив. Освоить зарезервированные слова, синтаксис описания. Ознакомиться с понятиями: индексации, тип элементов, длина массива, компонент массива, тип-диапазон, упорядоченность.

Теоретические положения

Любой из структурированных типов (а в Турбо Паскале их четыре: массив (лабораторная работа 6), запись (лабораторная работа 10), множество (лабораторная работа 10) и файл (лабораторная работа 9)) характеризуется множественностью образующих этот тип элементов, то есть переменная или константа структурированного типа всегда имеет несколько компонентов. Каждый компонент, в свою очередь, может принадлежать структурированному типу, что позволяет говорить о возможной вложенности типов. В Турбо Паскале допускается произвольная глубина вложенности типов, однако суммарная длина любого из них во внутреннем представлении не должна превышать 65520 байт.

Массивы

Массивы в Турбо Паскале во многом схожи с аналогичными типами данных в других языках программирования. Отличительная особенность массивов заключается в том, что все их компоненты – данные одного типа (возможно, структурированного). Эти компоненты можно легко упорядочить и обеспечить доступ к любому из них простым указанием его порядкового номера, например:

```
type
    digit = array [0..9] of char:
    matrix = array [byte] of single;
```

```
var
    m: matrix;
    d: digit;
    i: integer;
    ……
    m[t]:= ord(d[i – 1]) / 10;
    ……
```

Описание типа массива задается следующим образом:

<имя типа> = **ARRAY**[<список индексных типов>] **OF** <тип>

Здесь <имя типа> – правильный идентификатор;

ARRAY(массив), **OF**(из) – зарезервированные слова;

<список индексных типов> – список из одного или нескольких индексных типов, разделенных запятыми;

квадратные скобки, обрамляющие список, – требование синтаксиса;

<тип> – любой тип Турбо Паскаля.

В качестве индексных типов в Турбо Паскале можно использовать любые порядковые типы, кроме LONGINT и типов-диапазонов с базовым типом LONGINT.

Определить переменную как массив можно и непосредственно при описании этой переменной, без предварительного описании типа массива, например:

```
var
    a, b : array [1..101 of real;
```

Обычно в качестве индексного типа используется тип-диапазон, в котором задаются границы изменения индексов. Так как тип <тип>, идущий за словом OF, – любой тип Турбо Паскаля, то он может, в частности, быть и другим массивом, например:

```
type
    mat = array [0..5J of array [– 2..2] of array [char] of byte;
```

Такую запись можно заменить более компактной:

```
type
    mat = array [0..5, – 2..2, char] of byte;
```

Глубина вложенности структурированных типов вообще, а следовательно, и массивов – произвольная, поэтому количество элементов в списке индексных типов (размерность массива) не ограничено, однако суммарная длина внутреннего представления любого массива не может быть больше 65520 байт.

В Турбо Паскале можно одним оператором передать все элементы одного массива другому массиву того же типа, например:

```
var
    a, b: array[1..5] of single;
    .....................................
    a:= b;
```

После этого присваивания все пять элементов массива A получат те же значения, что и в массиве B. Однако над массивами не определены операции отношения. Нельзя, например, записать:

IF A = B **THEN** ...

Сравнить два массива можно только поэлементно.

Массив – это набор объектов одного типа, у каждого из которых есть индекс (номер). Например, элементы массива длины 20 могут иметь индексы 1, 2,..., 20 или 0, 1,..., 19. Способ индексации, тип элементов, длина массива фиксируются в определении того типа, к которому принадлежит массив.

Определение, имеющее вид mass1 = **ARRAY** [I.. 20] **OF** real, – это определение типа, имя которого mass1. Объектами типа mass1 будут упорядоченные наборы по 20 элементов, имеющих тип real; диапазон изменения значения индекса – от 1 до 20. Это определение типа, предваренное служебным словом **TYPE**, помещается в программу перед совокупностью описаний переменных.

Пусть переменная *a* описана в программе как переменная типа mass1:

VAR a: mass1;

Тогда при выполнении программы значениями переменной *a* будут массивы длины 20, элементы которых имеют тип real. Для того чтобы рассматривать эти элементы по отдельности, для них применяются обозначения *a*[1], *a*[2],…, *a*[20].

Переменная *a* – это переменная типа mass1, переменные *a*[1], *a*[2],…, *a*[20] – это переменные типа real. С переменными *a*[1], *a*[2],…, *a*[20] можно обращаться как с обычными переменными типа real (то есть, как с переменными-идентификаторами *x, y, z* и так далее). Заключенный в квадратные скобки индекс – это обязательно целочисленная величина, ею может быть произвольное выражение со значением любого порядкового типа BYTE, INTEGER, SHORTINT или WORD, но не LONGINT. Например, переменные *a*[*i*], *a*[2**i*],…, *a*[2**i*–1] удобно использовать для поочередного рассмотрения в цикле всех элементов массива, элементов, стоящих на четных и нечетных местах. Здесь проявляется преимущество обозначений *a*[1], *a*[2],…, *a*[20] перед обозначениями *a*1, *a*2,…, *a*20. Значение индекса обязано лежать в указанном в определении типа диапазоне, в данном случае – в диапазоне от 1 до 20.

Операции над объектами типа mass1 – это доступ к отдельным элементам массивов через индексы и изменение отдельных элементов массивов с помощью операций, связанных с типом real.

Пример.

Пусть *a*[1], *a*[2],…, *a*[20] – количество осадков в миллиметрах, выпадавшее в Москве в течение первых 20 лет нашего столетия. Надо вычислить среднее количество осадков и отклонение от среднего для каждого года.

```
type mas = array[1..20] of real;
var
   a:mas;
   i:integer;
   s:real;
begin
```

```
  s:=0;
  for i:=1 to 20 do begin
    read(a[i]);
    s:=s + a[i]
  end;
  s:= s / 20;
  writeln(s);
  for i:= 1 to 20 do writeln(s – a[i]);
end.
```

В определении типа mas указан диапазон изменения индекса от 1 до 20. Можно было бы взять диапазон от 0 до 19. Вообще, можно было в качестве границ взять любые целые числа, разность между которыми равна 19.

В квадратных скобках в качестве индекса можно помещать выражение со значением порядкового типа, например, можно писать *a*[2**i*+1]. Значение индекса не должно выходить из объявленного диапазона.

Матрицы. Массивы массивов

В определении типа u = **ARRAY** [n_1..n_2] **OF** T; в качестве базового типа T может выступать любой стандартный или ранее определенный тип. Это позволяет двумерный массив (матрицу) рассматривать как массив массивов.

Переменные с двумя индексами удобны для работы с таблицами.

В условиях задач на тему «Массивы» одномерный массив может быть обозначен:

1) A(*n*);

2) массив порядка *n*;

3) элементы массива $A = (a_1, a_2, \ldots, a_n)$;

4) элементов массива $a_1, a_2, \ldots, a_n$.

Для двумерных массивов все вышеперечисленные обозначения справедливы с добавлением второго индекса, то есть A(*n*, *m*), квадратная матрица порядка *n*, двумерный массив порядка *n*×*m*.

Все эти описания, при условии отсутствия специальных ограничений, желательно использовать явно (например: если в задаче данные о количестве осадков с 3 по 7 мая хранятся в массиве *A*, то, хотя для решения задачи важно только количество сохраняемых элементов, удобнее задать размерность, как [3.. 7]),чем сделать это неявно (например: [1..5]).

В описании могут быть сформулированы массивы A: array[1..n] of T;, где T – тип элементов (действительные, вещественные числа: REAL, SINGLE, DOUBLE, EXTENDED, или целые: SHORTINT, BYTE, INTEGER, WORD, LONGINT).

Уточним некоторые особенности работы с массивами в Турбо Паскале 7.0.

Часто при работе с массивами приходится заполнять его данными. Хорошо, когда можно это сделать через счетчик случайных чисел.

Пример.

```
var
  mas: array[1..5, boolean] of byte;
  i: byte;
  b: boolean;
begin
  randomize;
  for i:= 1 to 5 do begin
    for b:= false to true do begin
      mas[i,b]:= random(255);
      write(mas[i, b]: 10)
    end;
    writeln;
  end;
  writeln
end.
```

Однако бывает ситуация, когда необходимо, чтобы массив был заполнен определенными данными (например, в решении систем методом Гаусса необходимо обрабатывать вполне определенную матрицу). Тогда, чтобы не набирать при каждом запуске программы одни и те же значения, следует воспользоваться типизированным константа-массивом. Отличие типизированных констант от обычных в том, что они могут меняться в процессе выполнения программы.

Пример.

```
const
  mas: array[1..5, boolean] of integer =
                    ((1,2),(3,4),(5,6),(7,8),(9,0));
var
  i: byte;
  b: boolean;
begin
  for i:= 1 to 5 do begin
    for b:= false to true do
      write(mas[i, b]: 10);
      writeln;
  end;
  writeln
end.
```

Если при работе с массивом необходимо при каждом запуске менять вводимые данные, приходится их вводить вручную.

Пример.

```
var
  mas: array[1..5, boolean] of byte;
  i: byte;
  b: boolean;
begin
  randomize;
  for i:= 1 to 5 do
```

```
   for b:= false to true do begin
      write('Введите mas[',i,',',b,']');
      readln(mas[i, b])
   end;
   writeln;
   for i:= 1 to 5 do begin
     for b:= false to true do
      write(mas[i, b]: 10);
      writeln;
   end;
   writeln
end
end.
```

Уже было сказано, что на месте индекса можно использовать функции и выражения. Однако необходимо следить за соответствием типов.

Пример.

mas[4/2,false]:= 5; – появится сообщение об ошибке.

mas[round(4/2),false]:= 5; – ошибок нет.

Следующая особенность при работе с массивами, о которой следует напомнить, – правила использования идентичных типов.

Пример.

```
var
  mas, massiv: array[1..5, boolean] of integer;
```

Тогда разрешены следующие виды присваивания:

mas:= massiv;

mas[1]:= massiv[3];

massiv[4]:= mas[2];

mas[1]:= mas[5].

Говорят, что переменные имеют идентичный тип, если они имеют общее описание типа, то есть если бы в предыдущем примере переменные были описаны так:

mas: array[1..5, boolean] of integer;

massiv: array[1..5, boolean] of integer;

то компилятор при попытке переслать одной командой все значения из одного массива в другой выдал бы сообщение об ошибке.

Порядок выполнения работы

1. Уточните номер своего варианта.
2. Составьте блок-схему алгоритмов заданий варианта.
3. Напишите программу по составленной блок-схеме.
4. Отладьте программу (исключите все сообщения об ошибках) и подберите значения для тестирования программ (запишите результаты тестирования).
5. В текстовом редакторе WORD или рукописно оформите отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.
2. Задания варианта
3. Блок-схема алгоритмов заданий варианта.
4. Программные единицы.
5. Результат выполнения программы.
6. Описание использовавшихся операций копирования строк.

Варианты задач

	Вариант 1
1.	Информация о количестве выпадавших в течение месяца осадках задана в виде массива. Определить общее количество осадков за месяц. Число дней в месяце определять программно по названию.
2.	В матрице A(6, 6) заменить строку и столбец, содержащие минимальный элемент матрицы, на первую строку. Вывести на печать массив A(6, 6), до и после внесения изменений.
	Вариант 2
1.	Информация о температуре воздуха за месяц задана в виде массива. Определить, сколько раз температура опускалась ниже 0^oC. Число дней в месяце определять программно по названию.
2.	Написать программу, которая позволяет проверить, что в данной квадратной целочисленной матрице порядка суммы элементов во всех строках и всех столбцах равны между собой.

	Вариант 3
1.	Информация о среднесуточной температуре воздуха за месяц задана в виде одномерного массива (*t*(*dey*)). Определить, сколько дней температура была ниже среднемесячной. Число дней в месяце определять программно по названию.
2.	Получить целочисленную квадратную матрицу A(7), элементами которой являются числа 1, 2,..., 49, расположенные в ней по спирали.
	Вариант 4
1.	Сила и направление ветра на горном плато регистрируются один раз в день по очереди двумя исследователями. Каждый месяц результаты сводятся в таблицу. Составить программу, выполняющую эту операцию. Решение сводится к объединению двух массивов в один с чередованием элементов исходных массивов. Число дней в месяце определять программно по названию.
2.	Дана действительная квадратная матрица *B*(7). Найти последовательность действительных чисел b_1, b_2, … , b_{49}, получающуюся при чтении данных по спирали.
	Вариант 5
1.	Информация о количестве осадков, выпадавших в течение месяца, и о температуре воздуха задана в виде массивов. Определить, какое количество осадков выпало в виде дождя, какое – в виде снега. (Считать, что идет дождь, если температура воздуха больше 0^oC.)
2.	Дана действительная квадратная матрица порядка *n*, все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять местами его с элементом, стоящим на пересечении этих диагоналей.
	Вариант 6
1.	Рост учеников класса представлен в виде массива. Рост девочек кодируется знаком «+», рост мальчиков – знаком «–». Определить средний рост мальчиков.
2.	В данной действительной квадратной матрице *A* порядка *n* найти наибольший по модулю элемент. Получить квадратную матрицу *B* порядка $n - 1$ путем выбрасывания из исходной матрице *A* строки и столбца, на пересечении которых расположен элемент с найденным значением.
	Вариант 7
1.	В области 10 районов. Известны площади, засеиваемые пшеницей, и средняя урожайность (число центнеров с 1 га) в каждом районе. Площади, засеиваемые пшеницей, и урожайность для каждого района задаются в двух массивах. Определить количество пшеницы, собранное в области, и среднюю урожайность по области.
2.	Даны целочисленная матрица $A(n\times 3)$, целые числа k, t ($1 \le k \le n$, $1 \le t \le n$, $k \ne t$). Создать матрицу *B* так, чтобы элементы матрицы *A* в *B* расположились следующим образом: строка с номером *k* непосредственно следовала за строкой с номером *t*, сохранив порядок следования остальных строк.
	Вариант 8
1.	Ртутные термометры применяются для измерения температуры до $-39{,}4^oC$. Используя информацию о минимальной температуре, зафиксированной в каждый из последних 100 лет в г. Воронеже, определить, можно ли поставлять ртутные термометры в этот город.
2.	Даны две действительные квадратные матрицы *A* и *B* порядка *n*. Получить матрицу *C* прибавлением к элементам каждого столбца матрицы *A* произведения элементов соответствующих строк матрицы *B*.

	Вариант 9
1.	Фамилии участников соревнований по фигурному катанию после короткой программы расположены в порядке, соответствующем занятому месту. Составить список участников в порядке их стартовых номеров для произвольной программы (участники выступают в порядке, обратном занятым местам). Дополнительный массив не использовать.
2.	Даны две действительные квадратные матрицы A и B порядка n. Получить матрицу C умножением элементов каждой строки матрицы A на наибольшее из значений элементов соответствующей строки матрицы B.
	Вариант 10
1.	При поступлении в институт абитуриенты, получившие «двойку» на первом экзамене, ко второму экзамену не допускаются. Считая фамилии абитуриентов и их оценки после первого экзамена исходными данными, составить список допущенных ко второму экзамену.
2.	Дана действительная квадратная матрица $A(n)$. Преобразовать матрицу по правилу: строку с номером n сделать столбцом с номером n, а столбец с номером n сделать строкой с номером n. Дополнительную матрицу не использовать.
	Вариант 11
1.	Вычислить значения функции $z = \dfrac{\sin(x_i \cdot y_i)}{x_i - y_i}$, если y задано массивов (y_1, y_2, …, y_{30}), x изменяется начиная со значения 0 с шагом 0,1.
2.	Дана целочисленная матрица $A(6 \times 9)$. Не используя дополнительных матриц, получить матрицу, получающуюся из данной перестановкой строк: первой с последней, второй – с предпоследней и т. д.
	Вариант 12
1.	Вычислить значения функции $w = \dfrac{ax_i + by_i + cz_i}{3}$, если x изменяется начиная от 0 с шагом 0,1; y – начиная от введенного с клавиатуры y_0 с шагом 0,2; z задано массивом (z_1, z_2, …, z_{11}).
2.	Дана целочисленная матрица $A(6 \times 9)$. Найти матрицу B, получающуюся из A перестановкой столбцов: первого с последним, второго с предпоследним и т. д.
	Вариант 13
1.	Дан массив $A(50)$, заполненный случайными вещественными значениями, заполнить массив N значениями функции, вычисленными следующим образом: $n_i = \begin{cases} +1, x > 0 \\ 0, \ \ x = 0 \\ -1, x < 0 \end{cases}$.
2.	Даны действительные числа a_1, a_2, …, a_n, квадратная матрица A порядка n ($n \geq 6$). Получить матрицу B порядка $n \times (n + 1)$, вставив в исходную матрацу между пятым и шестым столбцами новый столбец с элементами a_1, a_2, …, a_n.
	Вариант 14
1.	Переписать положительные элементы массива $X = (x_1, x_2, \ldots, x_{16})$ подряд в массив Y. На экран вывести элементы массива Y.
2.	Даны целые числа a_1, a_2, …, a_{10}, целочисленная квадратная матрица A (n). Заменить нулями в матрице те элементы, для которых имеются равные среди a_1, a_2, …, a_{10}.
	Вариант 15
1.	Переписать положительные элементы массива $X = (x_1, x_2, \ldots, x_{90})$ в массив T, а отрицательные – в массив D. Элементы в массивах T и D следует располагать

	подряд.
2.	Даны действительная матрица размера $n \times n$, действительные числа a_1, a_2, …, a_n, b_1, b_2, …, b_{n+1}, натуральные числа p, q ($p \le n$, $q \le n + 1$). Образовать новую матрицу размера $(n + 1) \times (n + 1)$ вставкой после строки с номером p данной матрицы новой строки с элементами a_1, a_2, …, a_n, и последующей вставкой после столбца с номером q нового столбца с элементами b_1, b_2, …, b_{n+1}.
	Вариант 16
1.	Вычислить сумму положительных элементов массива $V = (v_1, v_2, \ldots, v_{90})$. В массиве есть положительные элементы.
2.	В матрице $A(6, 6)$ найти среднее арифметическое всех элементов и номер строки с наибольшим числом положительных элементов. Заменить элементы этой строки на первый элемент матрицы.
	Вариант 17
1.	Вычислить среднее арифметическое s элементов массива b_1, b_2, …, b_{60}, удовлетворяющих условию $0 \le b_i \le 1$. Если таких элементов нет, то считать $s = 0$.
2.	В матрице $A(6,6)$ найти среднее арифметическое элементов под главной диагональю матрицы и наименьший элемент всей матрицы. Если наименьший элемент находится в нижней треугольной матрице, заменить его на 1, если в верхней – заменить его на 0, если он на главной диагонали, его значение не менять.
	Вариант 18
1.	Подсчитать количество положительных, отрицательных и нулевых элементов массива $V = (v_1, v_2, \ldots, v_{90})$.
2.	В матрице $A(6,6)$ найти строку с наименьшим элементом и поменять ее со столбцом с наибольшим числом положительных элементов.
	Вариант 19
1.	Вычислить сумму элементов главной диагонали матрицы $A(20\times20)$.
2.	В матрице $A(6, 6)$ определить число отрицательных элементов в каждом столбце и записать их в дополнительную строку, а также найти среднее арифметическое элементов каждой строки и записать их в дополнительный столбец.
	Вариант 20
1.	Вычислить сумму элементов главных диагоналей матрицы $B(15\times15)$. Элементами i-й строки, расположенными на главных диагоналях матрицы, размерности 15×15, являются $b_{i,i}$ и $b_{i,16-i}$.
2.	В матрице $A(6, 6)$ определить сумму элементов, стоящих под главной диагональю и сумму элементов над главной диагональю. Заменить отрицательные элементы той части матрицы, в которой эта сумма меньше на 0.
	Вариант 21
1.	Вычислить среднее геометрическое элементов массива $G = (g_1, g_2, \ldots, g_{35})$, удовлетворяющих условию $G_I > d$. Среднее геометрическое вычислить по формуле $s = \sqrt[n]{p}$, где p – произведение из n сомножителей.
2.	В матрице $A(6, 6)$ определить число отрицательных элементов над главной диагональю. Если это число больше трети всех элементов матрицы, транспонировать матрицу K, если меньше, то оставить матрицу без изменений.
	Вариант 22
1.	Подсчитать количество элементов целочисленного массива $V = (v_1, v_2, \ldots, v_{90})$, кратных 7.
2.	В матрице $A(6, 6)$ найти строку с наименьшим числом положительных элементов и сумму этих положительных элементов.
	Вариант 23

1.	Найти наибольше**е** значение $(x_i - y_i)$ для массивов $X = (x_1, x_2, \ldots, x_{50})$, и $Y = (y_1, y_2, \ldots, y_{50})$.

2.	В матрице A(6, 6) найти среднее арифметическое всех элементов, строку с наименьшим числом элементов меньших среднего арифметического. Вывести на экран массив *A* и найденную строку.
	Вариант 24
1.	Найти численно наименьшее значение функции $y = \frac{\sin x}{1+x}$ и значение аргумента, при котором оно получено. Значение аргумента *x* менять от 0 до 10 с шагом 0,1.
2.	В матрице A(6, 6) найти строку с наименьшим числом положительных элементов и поменять ее местами с таким же столбцом. Вывести на печать массив до и после изменения.
	Вариант 25
1.	Найти и записать на место x_1 наибольший элемент, на место x_{60} – наименьший элемент массива $(x_1, x_2, \ldots, x_{60})$.
2.	В матрице O(7, 7) переставить местами строку с наибольшим элементом матрицы со строкой, среднее арифметическое элементов которой наименьшее.
	Вариант 26
1.	Найти наибольший элемент главной диагонали матрицы X (10×10) и вывести на печать всю строку, в которой он находится.
2.	В матрице A(6, 6) подсчитать число столбцов, среднее арифметическое элементов которых меньше среднего арифметического всех элементов матрицы.
	Вариант 27
1.	Определить количество положительных и количество отрицательных элементов матрицы W(16×18).
2.	В матрице A(6, 6) найти число всех элементов, меньших среднего арифметического элементов матрицы, и поменять местами первый столбец и первую строку. Вывести на печать массив до и после изменения.
	Вариант 28
1.	Найти наименьший элемент матрицы *A*(15, 20) и номер строки и столбца, в которых он находится.
2.	В матрице *A*(6, 6) найти строку с наименьшим средним арифметическим положительных элементов и строку с наибольшим числом положительных элементов. Поменять эти строки местами. Вывести на печать матрицу *A*(6, 6) до и после внесения изменений.
	Вариант 29
1.	В одномерном массиве B(12) поменять местами минимальный и первый положительный элемент. Вывести на печать массив B(12) до и после внесения изменений.
2.	В матрице A(6, 6) найти строку с наименьшим числом положительных элементов и строку с наибольшим числом положительных элементов. Поменять эти строки местами. Вывести на печать массив A(6, 6) до и после внесения изменений.
	Вариант 30
1.	В одномерном массиве B(12) заменить все элементы, меньшие среднего арифметического всех элементов, на значение первого элемента массива. Вывести на печать массив B(12) до и после внесения изменений.
2.	Переписать положительные элементы главной диагонали матрицы X(10, 10) в одномерный массив Y. Вывести на печать массивы X(10, 10) и Y.

Решение типового варианта

Рассмотрим пример оформления работы.

Пусть вариант студента 31.

	Вариант 31
1.	Дана последовательность символов (объектов типа char) s_1, s_2, …, s_{300}. Требуется определить, совпадает ли начальная часть s_1, s_2, …, s_{150} последовательности с ее концевой частью s_{151}, s_{152}, …, s_{300}.
2.	Написать программу, которая позволяет проверить, что в данной целочисленной матрице 17×17 суммы элементов во всех строках и всех столбцах равны между собой.

Сформируем отчет к лабораторной работе 6.

1. Титульный лист.

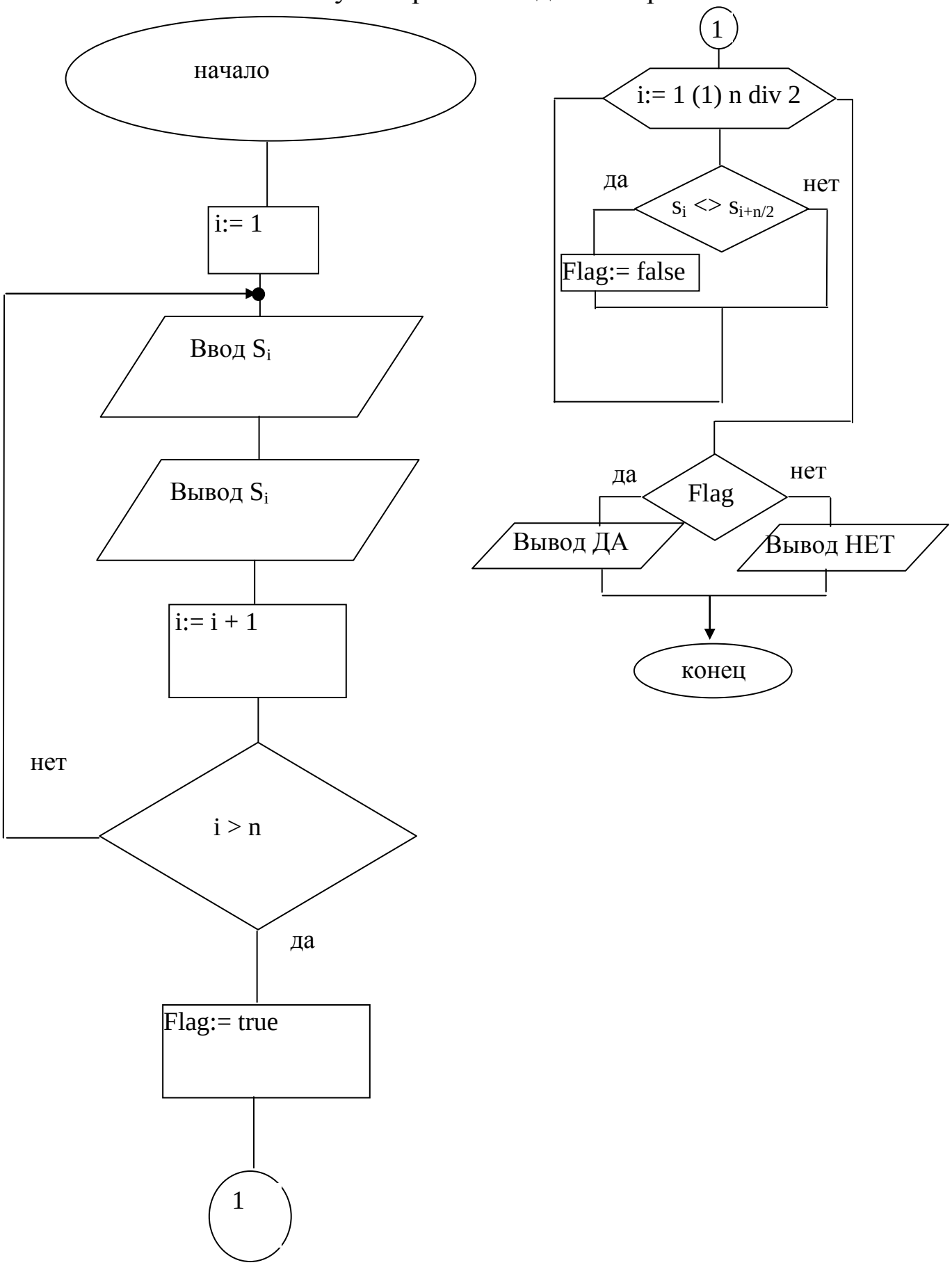
2. Сформируем задание варианта 31.

1. Дана последовательность символов (объектов типа char) s_1, s_2, …, s_{300}. Требуется определить, совпадает ли начальная часть s_1, s_2, …, s_{150} последовательности с ее концевой частью s_{151}, s_{152}, …, s_{300}.

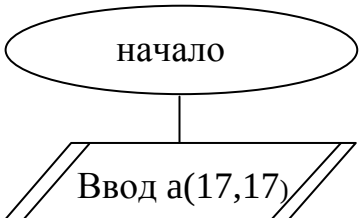
2. Написать программу, которая позволяет проверить, что в данной целочисленной матрице 17×17 суммы элементов во всех строках и всех столбцах равны между собой.

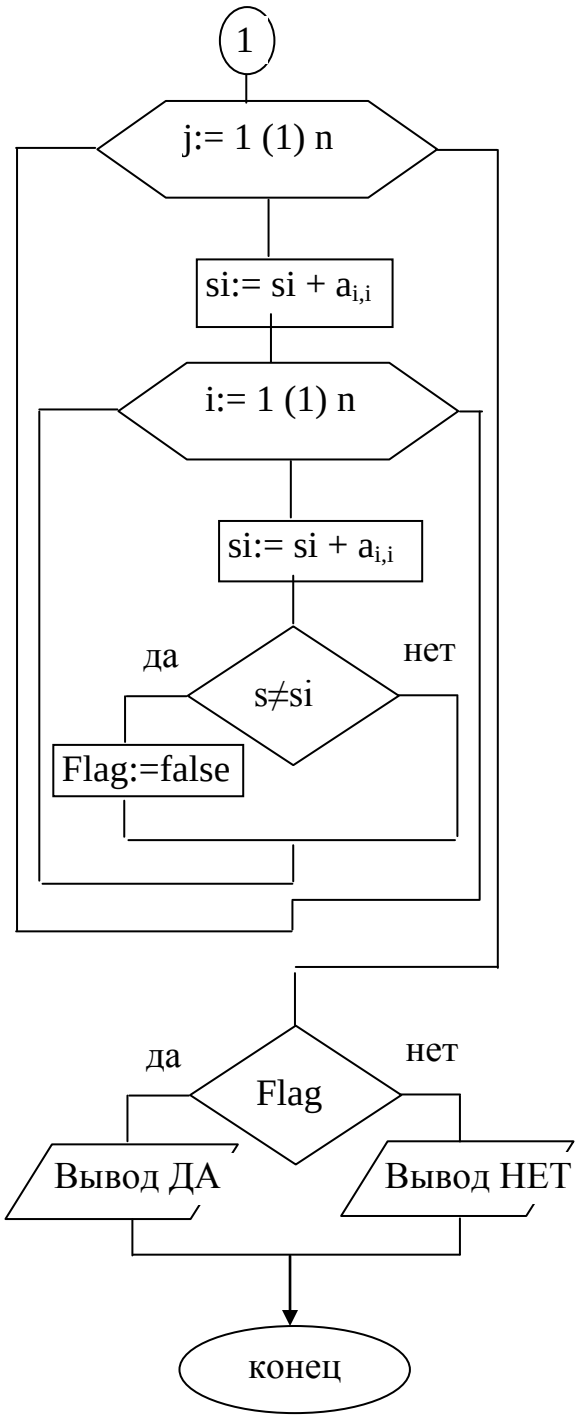
3. Составим блок-схему алгоритмов заданий варианта.

1.



2.





4. Напишем программу по составленной блок-схеме.

```
{1.}uses crt;
  var
    s: array[1..10] of char;
    i,j: integer;
    Flag:boolean;
  const
    n=10;{n = 300}
begin
  clrscr;
  write('Введите строку S:');
  i:=1;
  repeat
    s[i]:=readkey;
    write(s[i]);
    i:=i+1
  until i>n;
  writeln;
  Flag:=true;
  for i:=1 to n div 2 do
    if s[i]<>s[i+n div 2] then Flag:=false;
  if Flag then writeln('Совпадает') else
writeln('Не совпадает');
  readkey;
end.
{2.} uses crt;
const
  n=5;
type
  mas=array[1..n,1..n] of integer;
const
(*  a:mas=(( 3,16, 9,22,15), {Закомментировать}
          (20, 8,21,14, 2), { для случайных чисел}
         ( 7,25,13, 1,19), {весь текущий раздел}
```

```
          (24,12, 5,18, 6), {константа}
          (11, 4,17,10,23));*)
  a:mas=((-2, 2,-4,-1, 0), {Закомментировать для}
         ( 4, 1,-4,-5, 4), {случайных чисел}
         ( 1, 1, 4, 2, 1), {весь текущий раздел}
         ( 4,-1,-5,-1,-1), {константа}
         (-5,-2, 1,-4,-4));
var
  i,j:byte;
  Flag:boolean;
  s,si:integer;
{  a:array[1..n,1..n] of
integer;}{Раскомментировать для}
begin    {случайных чисел}
  clrscr;
  randomize;
  s:=0;
  for i:=1 to n do begin
    for j:=1 to n do begin
{      a[i,j]:=random(10)-5;}{Тоже}
      write(a[i,j]:3);
    end;
    writeln;
  end;
  Flag:=true;
  s:=0;
  for i:=1 to n do s:=s+a[1,i];
  for i:=1 to n do begin
    si:=0;
    for j:=1 to n do begin
      si:=si+a[i,j];
    end;
    if s<>si then Flag:=false;
```

```
  end;
  for j:=1 to n do begin
    si:=0;
    for i:=1 to n do begin
      si:=si+a[i,j];
    end;
    if s<>si then Flag:=false;
  end;
  if Flag then writeln('Верно') else writeln('Не
верно');
  readkey;
end.
```

5. Представленные программы не содержат ошибок, подберем значения для тестирования программ.

1. а) s = 1234512345 совпадают;
   б) s = 1231231231 не совпадают;
2. а) a=((3,16, 9,22,15),
   (20, 8,21,14, 2),
   (7,25,13, 1,19),
   (24,12, 5,18, 6),
   (11, 4,17,10,23)); верно
   б) a=((-2, 2,-4,-1, 0),
   (4, 1,-4,-5, 4),
   (1, 1, 4, 2, 1),
   (4,-1,-5,-1,-1),
   (-5,-2, 1,-4,-4)); не верно

6. В текстовом редакторе WORD оформим отчет в соответствии с содержанием.

Список литературы

1. Григас, Г. Начала программирования : кн. для учащихся : пер. с лит. / Г. Григас ; под. ред. Ю. А. Первина. – М. : Просвещение, 1987. – 112 с. : ил.

2. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

3. Попов, В. Б. Turbo Pascal для школьников : учебное пособие / В. Б. Попов. – 3-е доп. изд . – М. : Финансы и статистика, 1999. – 528 с.

4. Немюгин, С. А. Turbo Pascal : учебник / С. А. Немюгин. – СПб. : Питер, 2001. – 496 с.

Вопросы для самопроверки

1. Перечислите структурированные типы данных.

2. Дайте определение массива. Укажите служебные слова для описания массива. Покажите описание двумерного массива $B(2, 3)$ как массива массивов.

3. Дайте описание двумерного массива A(10,10) как одномерного массива одномерных массивов. $a_{i\ j}$ – целые значения в диапазоне от –100 до 100.

4. Дайте описание массива A(10), если a_i – целые значения в диапазоне от –100 до 100.

5. Дайте описание массива A(10), если a_i – вещественные значения порядка, не превышающего 10^{10}.

6. Дайте описание массива A(10), если a_i – строки, длиной не более пяти символов.

7. Напишите фрагмент программы, обеспечивающий заполнение одномерного массива A(25) значениями с клавиатуры.

8. Напишите фрагмент программы заполнения двумерного массива A(3,3) вещественными значениями, используя счетчик случайных чисел.

9. Задайте двумерный массив произвольных целых чисел (три строки, два столбца) как типизированный константа-массив.

10. Напишите функцию вычисления суммы элементов массива A(100).

11. Напишите фрагмент программы вывода двумерного массива A(3, 4) в виде прямоугольной таблицы.

12. Опишите переменные *a* и *b* как одномерные массивы идентичных типов с индексами от 1 до 10, предназначенные для хранения вещественных значений.

ЛАБОРАТОРНАЯ РАБОТА 7. ПОДПРОГРАММЫ ТУРБО ПАСКАЛЯ. ПРОЦЕДУРЫ И ФУНКЦИИ. ОБРАЩЕНИЕ К ПОДПРОГРАММАМ. ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ПОДПРОГРАММ В ЯЗЫКЕ ТУРБО ПАСКАЛЬ (2 ЧАСА)

Цель работы: описание процедур и функций. Обращение к подпрограммам. Параметры процедур и функций. Особенности использования подпрограмм в языке Паскаль. Опережающее описание. Побочные эффекты.

Теоретические положения

В лабораторной работе 4 описана форма повторяемости (цикличность), когда одни и те же действия многократно выполняются на одном этапе обработки информации. Исключительно широко распространена другая форма повторяемости, когда одна и та же последовательность действий должна выполняться на различных этапах обработки информации. В алгоритмах такого рода в различных местах встречаются фрагменты, одинаковые по выполняемым действиям и различающиеся только в значениях исходных данных. При составлении программы по такому алгоритму приходится задавать одну и ту же группу операторов, соответствующую каждому из повторяющихся фрагментов. Для более эффективного программирования подобных повторений в языке введено понятие подпрограммы. Повторяющаяся группа операторов оформляется в виде самостоятельной программной единицы – подпрограммы, записывается однократно, а в соответствующих местах программы обеспечивается лишь обращение к ней (ссылка).

Использование аппарата подпрограмм позволяет сократить объем и улучшить структуру программы с точки зрения наглядности и

читаемости, уменьшить вероятность ошибок и облегчить процесс отладки программы. Разложение программы на взаимосвязанные, но замкнутые и логически завершенные компоненты дает возможность выполнять разработку отдельных подпрограмм разным программистам и более или менее независимо друг от друга. Кроме того, подпрограмма может быть рассмотрена как самостоятельный блок (со своими входными и выходными данными), что позволяет использовать ее в общем иерархическом подходе при конструировании алгоритма и программы по принципам нисходящего проектирования.

В языке Паскаль подпрограммы реализуются в виде процедур и функций, которые вводятся в программу с помощью своего описания.

Описание процедур и функций

Процедуры описываются в разделе описательной части подпрограммы. Любая процедура состоит, аналогично программе, из заголовка процедуры и блока. Заголовок процедуры представляет собой:

PROCEDURE <ИМЯ> ((СПИСОК-ПАРАМЕТРОВ));

где **PROCEDURE** – служебное слово;

ИМЯ – имя процедуры, определяемое в соответствии с общими правилами построения идентификаторов;

СПИСОК-ПАРАМЕТРОВ – перечень имен для обозначения исходных данных и результатов работы процедуры с указанием их типов. Параметры, перечисленные в списке, называются формальными. Допускается описание процедуры, не содержащей формальных параметров:

PROCEDURE <ИМЯ>;

Содержательная часть процедуры представляет собой блок и состоит, следовательно, из раздела описаний (меток, констант, типов, переменных, процедур и функций) и раздела операторов, представляющего собой составной оператор BEGIN – END. Заканчивается блок процедуры точкой с запятой.

Пример.

Оформить в виде процедуры алгоритм вычисления степени $y = x^n$ с натуральным показателем *n*.

```
procedure step1 (n: integer; x: real; var y: real);
    var i: integer;
begin
    y: = l;
    for i := 1 to n do
        y: = y * x
end;
```

В заголовке процедуры с именем step1 перечислены параметры *n, x*, определяющие исходные данные процедуры, и параметр *y*, обозначающий значение искомой степени – результат выполнения процедуры. Указан также тип всех формальных параметров.

Тело процедуры (блок) состоит:

1) из описательной части, где определена переменная *i*, необходимая и имеющая смысл только внутри данной процедуры и называемая локальной переменной (значение локальной переменной недоступно в основной программе);

2) из составного оператора **BEGIN-END**, реализующего алгоритм вычисления степени действительного числа с натуральным показателем.

Пример.

Оформите алгоритм вычисления степени $y = x^n$ в виде процедуры без параметров:

```
procedure step2;
    var i: integer;
begin
    y: = l;
    for i: = 1 to n do
        y:=y * x
end;
```

В этом случае процедура STEP2 не содержит списка формальных параметров и работает с локальной переменной *i*, описанной в блоке процедуры, и переменными *x, n, y*, которые должны быть описаны в программе, содержащей описание данной процедуры. Пере-

менные *x, n, y* называются глобальными по отношению к процедуре step2. Значения глобальных переменных доступны и могут быть использованы в любой точке основной программы (в частности, внутри данной процедуры).

Функция – это подпрограмма, результат выполнения которой есть единственное скалярное значение, присваиваемое имени этой функции. Следовательно, функции являются частным случаем процедур и принципиально отличаются от них тем, что, во-первых, результат выполнения функции – одно значение, а процедуры – одно, ни одного или несколько; во-вторых, результат выполнения функции передается в основную программу как значение имени этой функции, а результаты выполнения процедуры – как значения ее параметров.

Описание функции аналогично описанию процедуры и состоит из заголовка и блока. Заголовок функции имеет вид:

FUNCTION <ИМЯ> ((СПИСОК-ПАРАМЕТРОВ)) : (ТИП);

где **FUNCTION** – служебное слово;

ИМЯ – имя функции;

СПИСОК-ПАРАМЕТРОВ – перечень формальных параметров (исходных данных) с указанием их типов;

ТИП – тип результата: значение, которое должно приобретать имя функции.

Допускается описание функции без параметров:

FUNCTION <ИМЯ> : <ТИП>;

В содержательной части подпрограммы-функции имени функции должно быть присвоено некоторое значение (значение ответа), то есть имя хотя бы один раз должно присутствовать в левой части некоторого оператора присваивания.

Пример.

Оформите в виде функции алгоритм вычисления степени $y = x^n$:

```
function step3 (n : integer; x : real): real;
    var i: integer; y:real;
begin
```

```
    y: = 1;
    for i : = 1 to n do
        y: = y * x;
    step3:=y
end;
```

В заголовке функции с именем step3 перечислены параметры *n*, *x*, определяющие ее исходные данные. Результат выполнения функции (значение локальной переменной *y*) присваивается ее имени step3. Тип результата (тип функции) – real, который указывается в заголовке функции при ее описании.

Введение локальной переменной *y* обязательно, так как использование имени функции в данной ситуации невозможно.

Обращение к подпрограммам

Описание процедуры (или функции), расположенное в разделе описаний, само по себе никакого действия не вызывает. Чтобы исполнить процедуру (или функцию), необходимо в нужном месте программы поместить обращение к ней. Обращение к процедуре производится с помощью специального оператора вызова процедуры или оператора процедуры, имеющего вид:

(ИМЯ) ((СПИСОК-АРГУМЕНТОВ));

где ИМЯ – имя процедуры, к которой происходит обращение;
СПИСОК-АРГУМЕНТОВ – перечень конкретных значений (выражений) и имен, подставляемых на место формальных параметров процедуры при ее выполнении.

При вызове процедуры формальные параметры, указанные в ее заголовке, заменяются аргументами в порядке их следования: первому слева параметру в списке ставится в соответствие первый аргумент, второму – второй и так далее. Аргументы, перечисленные в операторе процедуры, называются также фактическими параметрами. Число и тип формальных и фактических параметров должны совпадать.

Очень важно понимать суть и механизм замены формальных параметров фактическими. Формальные параметры – это переменные, фиктивно (формально) присутствующие в процедуре и определяющие тип и место подстановки фактических параметров. Фактические параметры – это реальные объекты программы, заменяющие в теле процедуры при ее вызове формальные параметры. Над этими объектами и производятся действия, предусмотренные операторами тела процедуры.

Если осуществляется вызов процедуры без параметров, то оператор процедуры представляет собой указание только имени этой процедуры.

Понятие глобальных и локальных переменных введено в языке Паскаль из-за наличия в нем понятия блочной структуры. Любая программа, процедура и функция представляют собой блок со своей областью описаний и могут содержать внутри этого блока описания других процедур и функций, а также обращения к ним. Программа и совокупность описанных в ней процедур и функций образуют блочную структуру. Блок, содержащий в своем разделе описаний другой блок (процедуру или функцию), называется внешним по отношению к нему. Блок, содержащийся в разделе описаний некоторого блока, называется внутренним или подблоком. Объекты, описанные внутри какого-либо подблока, являются по отношению к нему локальными и недоступны внешним блокам, то есть на них можно ссылаться только внутри блока, в котором они описаны.

Объекты, описанные в некотором внешнем блоке (или программе), доступны и могут быть использованы в любом его подблоке, то есть они являются глобальными по отношению к этим подблокам. Таким образом, объекты, локальные для некоторого блока, являются глобальными для всех его подблоков. Под этими объектами подразумевают метки и имена констант, типов, переменных, процедур и функций.

Практика программирования рекомендует выбирать имена локальных и глобальных переменных так, чтобы они не совпадали, иначе в программе могут возникнуть ошибки, связанные с путаницей имен. Совпадение имен формальных и фактических параметров подобных неприятностей в программах не вызывает, но целесообразно выбирать их тоже различными, что сделает программу более наглядной.

Обращение к функции осуществляется аналогично обращению к стандартным функциям (SIN, COS, SQRT и тому подобное, см. лабораторную работу 1) и является разновидностью операнда в выражениях в отличие от вызова процедуры, являющегося разновидностью оператора. В том месте выражения, где это необходимо, записывается имя функции, вслед за которым в скобках перечисляются фактические параметры. Если вызывается функция без параметров, то указывается только ее имя.

Параметры процедур и функций

При описании процедуры или функции в ее заголовке могут быть указаны параметры четырех видов:

1) параметры-значения;

2) параметры-переменные;

3) параметры-процедуры;

4) параметры-функции.

Параметры-значения используются для передачи исходных данных в подпрограмму (процедуру или функцию), в списке формальных параметров перечисляются через запятую с обязательным указанием их типов, например:

PROCEDURE PRMI (I, J: INTEGER; R, Z: REAL);

FUNCTION PRM2 (I, J: INTEGER; R15: REAL) : REAL;

В качестве соответствующего фактического параметра может быть использовано любое выражение идентичного типа, в частном случае константа или переменная. При вызове процедуры или функции фактические параметры вычисляются и используются как начальные значения формальных параметров, то есть осуществляется

подстановка значений. В процессе выполнения подпрограммы формальные параметры могут изменяться, но это никак не отразится на соответствующих фактических параметрах-переменных, которые сохраняют те значения, которые имели до вызова подпрограммы. Поэтому параметры-значения нельзя использовать для передачи результатов из подпрограммы в основную программу.

Параметры-переменные используются для определения результатов выполнения процедуры и в списке формальных параметров перечисляются после служебного слова **VAR** с обязательным указанием типа:

PROCEDURE PRM3 (**VAR** K, L: INTEGER; **VAR** Z: REAL);

В заголовке функции параметры-переменные использовать не рекомендуется, так как если результатов выполнения подпрограммы несколько, то целесообразнее применить процедуру.

Фактические параметры, соответствующие формальным параметрам-переменным, могут быть только переменными (не выражениями) того же типа. При обращении к процедуре и замене формальных параметров фактическими осуществляется передача адреса переменной (фактического параметра), а не ее значения. В результате все операторы процедуры манипулируют непосредственно с переменной-фактическим параметром, то есть все изменения формальных параметров относятся и к фактическим.

В качестве формальных параметров в языке Паскаль допустимо использование также имен процедур и функций.

Параметры-процедуры в списке формальных параметров указываются после служебного слова **PROCEDURE**:

PROCEDURE PRM4 (I, J: INTEGER; VAR Z: REAL; **PROCEDURE** FF);

Параметры-функции в списке формальных пар указываются после служебного слова **FUNCTION** с указанием типа функции:

PROCEDURE PRM5 (I, J: INTEGER; VAR Z: REAL; **FUNCTION** FG: REAL);

При вызове подпрограммы на место формальных параметров-процедур и параметров-функций осуществляется подстановка имен соответствующих фактических процедур или функций. При этом если процедуры и функции, фигурирующие в качестве формальных параметров, имеют, в свою очередь, параметры, то они могут быть только параметрами-значениями.

Формальные параметры могут указываться в заголовке процедуры и функции в любом порядке, а параметры одного типа не обязательно группировать в одном месте. Однако лучше придерживаться какой-то одной (установленной для себя) формы их записи.

Следующий пример показывает конструкцию, позволяющую выбрать для вычисления не только аргумент функции, но и саму функцию. В конструкции использовано новое служебное слово FAR, которое позволяет компилировать программу в расчете на дальнюю модель памяти. Эта стандартная директива ставится сразу за заголовками процедур или функций, которые будут передаваться в качестве фактических параметров вызова.

Пример.

```
uses crt;
var
  s: string;
  a, i: byte;
  x: integer;
type
  func = function (x: integer): real;
function f1(x: integer): real; far;
begin
  f1 := sin(x)
end;
function f2(x: integer): real; far;
begin
  f2 := cos(x)
end;
function f3(x: integer):real; far;
begin
```

```
  f3 := sqr(x)
end;
function f4(x: integer): real; far;
begin
  f4 := sqrt(x)
end;
procedure tabl(x: integer; f: func);
begin
    writeln(f(x));
end;
begin
  clrscr;
  writeln('function f1', 'sin(x)');
  writeln('function f2', 'cos(x)');
  writeln('function f3', 'sqr(x)');
  writeln('function f3', 'sqrt(x)');
  writeln('wwod x i № fynction');
  repeat
    readln(i,x);
  until (i = 1) or (i = 2) or (i = 3) or (i = 4);
  case i of
  1: tabl(x, f1);
  2: tabl(x, f2);
  3: tabl(x, f3);
  else tabl(x, f4);
  end;
  readkey
end.
```

Особенности использования подпрограмм в языке Паскаль

Рекурсии. В языке Паскаль процедуры и функции могут быть рекурсивными. Процедура (функция) называется рекурсивной, если она вызывает саму себя. Такой вызов процедур (функций) может возникнуть вследствие рекурсивного описания. Под рекурсивным опи-

санием понимают следующее: в исполняемой части блока процедуры или функции присутствует обращение к ней самой.

Пример.

Составьте программу с рекурсивной функцией вычисления факториала:

```
Function fact (n: integer): integer;
Begin
    if n = 1 then fact:=1
        else fact: = n*fact(n – 1)
End;
BEGIN
    Writeln('Введите n:');
    Writeln(fact(n)
END;
```

При рекурсивном описании необходимо наличие базовой части описания, которая обеспечивала бы завершение рекурсивных вызовов функции (процедуры). В приведенном примере базовой частью, гарантирующей, что в конечном итоге будет достигнута ситуация, при которой fact (*n*) не зависит от fact(*n* – 1), является определение функции fact = 1 при *n* = 1.

Использование рекурсивных процедур и функций делает программу в целом более гибкой и наглядной, но не всегда достаточно эффективной. Довольно часто более эффективным является итерационное решение (с помощью итерационных циклов), что касается, в частности, задачи вычисления факториала. Рекурсии следует применять только там, где нет очевидного итерационного решения. Уровень вложенности рекурсий может быть ограничен в конкретных реализациях языка.

Опережающее описание. Если подпрограмма непосредственно вызывает саму себя, то такая рекурсия является простой рекурсией. Вполне возможна ситуация, когда подпрограмма вызывает одну или несколько промежуточных подпрограмм, которые, в конце концов,

вызывают первую подпрограмму. Такая рекурсия называется косвенной рекурсией. При таком взаимном обращении подпрограмм друг к другу нет возможности выполнить необходимое правило языка Паскаль, что все процедуры и функции должны быть определены (описаны) до их вызова. В этом случае используется опережающее описание процедур и функций, заключающееся в следующем: подпрограмма содержит описание только своего заголовка, вслед за которым ставится стандартное имя **FORWARD**; описание текста подпрограммы помещается далее (в любом месте раздела описаний процедур и функций) без повторения списка формальных параметров. Например:

```
var
a.в: real;
procedure p(x:real); forward;
procedure r(y;real);
begin
 . . . . . .
p(a);
. . . . . .
end;
procedure p;
begin
. . . . . .
r(b);
. . . . . .
end;
begin
. . . . . .
r(a); p(b);
. . . . . .
end.
```

Побочные эффекты. Побочным эффектом называется непредусмотренное искажение данных в подпрограмме. В языке Паскаль при использовании процедур и функций возможны следующие побочные эффекты: искажение подпрограммой исходных данных, передаваемых в нее; искажение подпрограммой глобальных переменных; использование параметра-переменной внутри функции и присвоение ему какого-либо значения.

Появление и действие побочных эффектов контролируется только программистом. Следует всячески избегать перечисленных нестандартных приемов программирования. Для этого необходимо внимательно разделять все параметры подпрограмм на параметры-значения (для исходных данных) и параметры-переменные (для результатов), не использовать (по возможности) в подпрограммах глобальных переменных, не использовать в функциях параметров-переменных, а, если результатов выполнения подпрограммы должно быть несколько, использовать процедуры. Особого внимания требует применение процедур и функций без параметров, работающих с локальными и глобальными переменными. Прибегать к ним следует лишь в особых случаях.

Порядок выполнения работы

1. Уточните номер своего варианта.
2. Составьте блок-схему алгоритмов заданий варианта.
3. Напишите программу по составленной блок-схеме.
4. Отладьте программу (исключите все сообщения об ошибках) и подберите значения для тестирования программ (запишите результаты тестирования).
6. В текстовом редакторе WORD или рукописно оформить отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.

2. Задания варианта
3. Описание использованных в работе операций.
4. Программные единицы.
5. Результат выполнения программы.
6. Описание использовавшихся операций копирования строк.

Варианты задач

	Вариант 1.
1.	ДАНЫ ДЕЙСТВИТЕЛЬНЫЕ ЧИСЛА *X, Y*. СОСТАВИТЬ ПРОГРАММУ ДЛЯ ВЫЧИСЛЕНИЯ ЗНАЧЕНИЙ ФУНКЦИИ $z(x,y)=\frac{2x^3-4x^2+x+1}{\sqrt{6y^3+y^2+2y}}+3Y^6+Y^4+2Y^2-6$. ВЫЧИСЛЕНИЕ МНОГОЧЛЕНА ОСУЩЕСТВЛЯТЬ В ПОДПРОГРАММЕ ПО СХЕМЕ ГОРНЕРА.
2.	Даны матрицы A (15, 20) и B (15, 30). Составить программу для нахождения сумм элементов каждой строки каждой матрицы, используя подпрограмму.
3.	Пусть $x_0 = 1$, $x_k=\frac{x_{k-1}^3}{5}$, $k = 1, 2, ...$ Найти первый член x_n, для которого $\lvert x_n - x_{n-1}\rvert < 10^{-5}$. Использовать рекурсивную подпрограмму.
	Вариант 2
1.	Даны целые числа n ≥ 0, m ≥ 0. Составить программу для вычисления числа сочетаний $C_n^m=\frac{n!}{m!(n-m)!}$. Использовать подпрограмму.
2.	Дана матрица A (15, 20). Составить программу для замены элементов каждой строки на сумму текущей и следующей строк. Использовать подпрограмму.
3.	Даны положительные действительные числа *a*, *x*, ε. В последовательности y_1, y_2,... , образованной по закону $y_0 = a$, $y_i=\frac{1}{2}\left(y_{i-1}+\frac{x}{y_{i-1}}\right)$, $i = 1, 2, ...$, найти первый член y_n, для которого выполнено неравенство $\left\lvert y_n^2 - y_{n-1}^2\right\rvert < \varepsilon$. Использовать рекурсивную подпрограмму.
	Вариант 3
1.	Составить программу для определения среднего балла группы по результатам сессии. Оценки групп сведены в матрицы A (25, 5), B (23, 5), C (22, 5), D (24, 5), где первый индекс – количество студентов в группе, второй – количество экзаменов текущей сессии. Средний балл вычислять в подпрограмме.
2.	Даны потоки данных a_i, b_j, где $i = 1, 2,..., 20$, $j = 1, 2,...30$. Составить программу для вычисления значений функции $z=\frac{\sum_{i=1}^{20}\lvert a_i\rvert+\sum_{i=1}^{30}\lvert b_i\rvert}{\sum_{i=1}^{20}a_i}$. Вычисление суммы выполнить в подпрограмме.
3.	Пусть $x_1 = x_2 = x_3 = 1$, $x_i = x_{i-1} + x_{i-3}$, $i = 4, 5$. Найти $\sum_{i=1}^{5}\frac{x_i}{2^i}$. Использовать рекур-

	сивную подпрограмму.
	Вариант 4
1.	Даны матрицы A(10, 12), B(15, 10) и C(8, 10). Составить программу для нахождения наименьших элементов и номеров строк и столбцов, в которых они находятся, используя подпрограмму.
2.	Даны действительные числа *a*, *x*, *y*. Составить программу для вычисления $z = \frac{\log_a 5x \cdot \log_5 (y+5)}{\log_{a+5}(xy)}$, используя подпрограмму. Формула $\log_a b = \frac{\log_c b}{\log_c a}$.
3.	Пусть $a_0 = 1$, $a_k = k \cdot a_{k-1} + \frac{1}{k}$, k = 1, 2, ... Дано натуральное число *n*. Получить a_n. Использовать рекурсивную подпрограмму.
	Вариант 5
1.	Даны массивы A(40), B(50), C(30) действительных чисел. Составить программу для записи в массив Z положительных элементов подряд, используя подпрограмму общего вида.
2.	Даны действительные числа *x*, *y*. Составить программу для вычисления значения $z = \frac{\sin^3 x - \cos^2 y}{\sqrt[3]{\sin x} + \sqrt{\lvert\cos y\rvert}}$, используя подпрограмму.
3.	Пусть $a_1 = u$, $b_1 = v$, $a_k = 2b_{k-1} + a_{k-1}$, $bk = 2a^2_{k-1} + b_{k-1}$, $k = 2, 3, \ldots$. Даны действительные *u*, *v*, натуральное *n*. Найти $\sum_{k=1}^{n} \frac{a_k b_k}{(k+1)!}$. Использовать рекурсивную подпрограмму.
	Вариант 6
1.	Даны вещественные числа *a*, *b*, *c*, *d*, *e*, *f*. Составить программу для вычисления значения функции $z = \frac{x_1 + y_1}{x_2 + y_2}$, где x_1, x_2 – корни уравнения $ax^2 + bx + c = 0$, y_1, y_2 – корни уравнения $dy^2 + ey + f = 0$. Если корни хотя бы одного уравнения комплексные, то вычисляется значение функции $z = \frac{x_1 + x_2}{y_1 + y_2}$. Для вычисления корней уравнения использовать подпрограмму общего вида.
2.	Составить программу для вычисления значения функции $z = \arcsin^3\left(\frac{1}{x}\right) + \arcsin^2 x + \arcsin\left(\frac{x^2}{x^2+1}\right)$, используя функцию. Учесть, что аргумент arcsin *x* не может быть по модулю больше 1.
3.	Пусть $v_1 = v_2 = 0$, $v_3 = 1.5$, $v_i = \frac{i+1}{i^2+1} \cdot v_{i-1} - v_{i-2} \cdot v_{i-3}$, $i = 4, 5, \ldots$ Дано натуральное *n* (n ≥ 4). Получить v_n. Использовать рекурсивную подпрограмму.
	Вариант 7
1.	Дан поток данных a_i, где $i = 1, 2, \ldots, 50$, Составить программу для вычисления значения выражения $z = \frac{1}{45} \cdot \left(\sum_{i=1}^{50} \sin^2 a_i + \sum_{i=1}^{50} \cos^2 a_i \right)$. Для вычисления сумм использовать подпрограмму-функцию.
2.	Даны натуральные числа *x*, *y*. Составить программу для вычисления значения $z = \frac{\log_3 x^2}{\log_x^2 y - \log_{x+3}(x^2+y)}$, используя функцию. Формула $\log_a b = \frac{\log_c b}{\log_c a}$.

3.	Пусть $x_0 = q$, $x_1 = q - 1$, $xk = q \cdot x_{k-1} + q \cdot x_{k-2} + q^{-1}$, $k = 2, 3, \ldots$ Дано действительное q, натуральное n ($n \geq 2$). Получить x_n. Использовать рекурсивную подпрограмму.
	Вариант 8
1.	Даны натуральные числа x, y. Составить программу для вычисления значения функции $z(x, y) = \dfrac{\log_2 x + \log_x y}{\log_{y+2}(x+y)}$, используя функцию. Формула $\log_a b = \dfrac{\log_c b}{\log_c a}$.
2.	Дана матрица A(80, 70). Составить программу для нахождения количества элементов в каждой строке больших среднего арифметического всей матрицы. Использовать подпрограмму.
3.	Пусть $a_1 = a_2 = 1$, $a_i = a_{i-2} + \dfrac{a_{i-1}}{2^{i-1}}$, $i = 2, 3, \ldots$ Найти произведение $a_1 \cdot a_2 \cdot \ldots \cdot a_{14}$. Использовать рекурсивную подпрограмму.
	Вариант 9
1.	Даны вещественные числа a, b, c, d. Составить программу, определяющую наибольшее из четырех чисел. Написать и использовать специальную функцию max2, находящую наибольшее из двух чисел.
2.	Даны действительные числа x, y. Составить программу для вычисления $z(x, y) = \arccos^2 \dfrac{x-5}{x^2-5} + \arccos^5 \dfrac{y^2+1}{y^3+1} + \arccos \dfrac{x+y^2-5}{x+y^3-5}$, используя функцию. Учесть, что аргумент arcsin x не может быть по модулю больше 1.
3.	Пусть $u_0 = u_1 = 0$, $u_i = \dfrac{u_{i-1} - u_{i-2}}{1 + u_{i-1}^2}$. Дано натуральное n ($n \geq 3$). Получить v_n. Использовать рекурсивную подпрограмму.
	Вариант 10
1.	Составить программу для вычисления приближенного значения биномиального ряда $(1+x)^m = 1 + \sum_{n=1}^{\infty} \dfrac{m(m-1)\cdots(m-n+1)\cdot x^n}{n!}$, при заданных значениях x – действительное число, n – натуральное, используя функцию для вычисления очередного слагаемого в разложении ряда.
2.	Дано действительное число x. Составить программу для вычисления $z = \underbrace{x + \sqrt{\lvert x \rvert} + \sqrt[3]{x} + \sqrt[4]{\lvert x \rvert} + \ldots}_{n}$, используя функцию.
3.	Пусть $a_1 = b_1 = 1$, $a_k = \dfrac{1}{2}\left(\sqrt{b_{k-1}} + \dfrac{1}{2}\sqrt{a_{k-1}}\right)$, $b_k = 2a^2_{k-1} + b_{k-1}$, $k = 2, 3, \ldots$ Дано натуральное n. Найти $\sum_{k=1}^{n} a_k b_k$. Использовать рекурсивную подпрограмму.
	Вариант 11
1.	Составить программу для вычисления функции $z = \dfrac{x_1 + x_1 y_1 + y_1}{x_1 + y_1}$, где x_1 – больший корень квадратного уравнения $x^2 - 4x - 1 = 0$; y_1 – больший корень квадратного уравнения $2y^2 + y - 1 = 0$. Корень уравнения вычислить, используя функцию.
2.	Даны действительные числа x, y. Составить программу для вычисления $z = \dfrac{\operatorname{arcctg}^3(\sin x + 1) + \operatorname{arcctg}^3 y + \operatorname{arcctg}(\operatorname{tg} x + 1)}{\operatorname{arcctg}(x^2 + y^2)}$, используя подпрограмму-

	функцию.
3.	Пусть $x_1 = y_1 = 1$, $x_i = 0{,}3 \cdot x_{i-1}$, $yi = x_{i-1} + y_{i-1}$, , i – 2, 3, ... Дано натуральное n. Найти $\sum_{i=1}^{n} \frac{x_i}{1+\lvert y_i \rvert}$. Использовать рекурсивную подпрограмму.
	Вариант 12
1.	Даны массивы A(8), B(7), C(10) действительных чисел. Составить программу для нахождения средних арифметических и средних геометрических значений результатов экспериментов, сведенных в эти массивы, используя подпрограмму.
2.	Дана матрица A(6, 5). Составить программу для нахождения количества элементов в каждом столбце меньших среднего арифметического всей матрицы. Использовать подпрограмму.
3.	Пусть $a_1 = b_1 = 1$, $a_k = 3_{k-1} + 2a_{k-1}$, $b_k = 2a^2_{k-1} + b_{k-1}$, k = 2, 3, ... Дано натуральное n. Найти $\sum_{k=1}^{n} \frac{k+2}{\left(1+a_k^2+b_k^2\right)\cdot k}$. Использовать рекурсивную подпрограмму.
	Вариант 13
1.	Даны двумерные массивы X(5, 6), Y(7, 4), Z(5, 7). Составить программу для нахождения максимальных элементов матриц. Результаты записать в массив P(3). Для нахождения максимального элемента матрицы использовать подпрограмму.
2.	Даны натуральные числа m и n. Вычислить значение выражения $\frac{m!+n!}{(m+n)!}$.
3.	Пусть $x_0 = 1$, $x_k = \frac{2}{5 - x_{k-1}^3}$, k = 1, 2, ... Найти первый член x_n, для которого $\lvert xn - x_{n-1} \rvert < 10^{-5}$. Использовать рекурсивную подпрограмму.
	Вариант 14
1.	Даны целочисленные массивы A (40), B (30), C (60). Составить программу для нахождения элементов этих массивов, кратных 3, используя подпрограмму.
2.	Дано натуральное число x. Вычислить $\sum_{s=0}^{10} \frac{1}{s\,(x+s)} \cdot \left(\frac{x}{2}\right)^{2s+x}$.
3.	Даны положительные действительные числа x, ε. В последовательности y_0, y_1, y_2,..., , образованной по закону $y_0 = x$, $y_i = \frac{1}{2}\left(y_{i-1} + \frac{x}{y_{i-1}}\right)$, i = 1, 2,..., найти первый член y_n, для которого выполнено неравенство $\lvert y^2_{\text{n}} - y^2_{\text{n-1}} \rvert < \varepsilon$. Использовать рекурсивную подпрограмму.
	Вариант 15
1.	Составить программу для вычисления и запоминания в массиве U(3) больших корней квадратных уравнений: $x^2 - 2x - 4 = 0$, $y^2 - 3y + 4 = 0$, $2z^2 + 5z - 1 = 0$. Корни уравнения искать в подпрограмме.
2.	Составить программу для вычисления значения $z = \underbrace{(x+1)\cdot\sqrt{\lvert x+1 \rvert}\cdot\sqrt[3]{x+1}\cdot\sqrt[4]{\lvert x+1 \rvert}\cdot\ldots}_{n}$ используя функцию. Учесть, что x может иметь отрицательные значения.
3.	Пусть $x_1 = x_2 = x_3 = 1$, $x_i = x_{i-1} + x_{i-3}$, $i = 4{,}5{,}\ldots$ Найти $\sum_{\text{i}=1}^{10} \frac{\text{x}_\text{i}}{2^\text{i}}$. Использовать рекурсивную подпрограмму.

	Вариант 16
1.	Даны действительные числа x, y. Составить программу для вычисления функции $z(x,y)=\dfrac{7x^4+x^3-5x^2+3x+10}{(5x^3-2x^2+4x+1)(2y^5+7y^4+y^3+8)}$. Вычисление значения многочлена осуществлять в подпрограмме по схеме Горнера.
2.	Дано натуральное число n. Вычислить $\dfrac{(-1)^{[\lg 1]}}{1}+\dfrac{(-1)^{[\lg 2]}}{2}+\ldots+\dfrac{(-1)^{[\lg n]}}{n}$, если обозначение [$a$] – означает «целая часть действительного числа a».
3.	Пусть $a_0 = 1$, $a_k = k \cdot a_{k-1} + \dfrac{1}{k}$, $k = 1, 2, \ldots$ Дано натуральное число n. Получить a_n. Использовать рекурсивную подпрограмму.
	Вариант 17
1.	Даны целые числа $n \le 10$, $m \le 10$, $k \le 10$, вещественные массивы A(n), B (m), C(k). Составить программу для вычисления и запоминания в массиве D(3) наибольших элементов массивов, используя подпрограмму-функцию.
2.	Даны вещественные числа x, y. Составить программу для вычисления значения $z=\dfrac{1}{\log_{x+3}(x^2+y)\log_y^3(x+y)^2}$, используя функцию. Формула $\log_a b=\dfrac{\log_c b}{\log_c a}$.
3.	Пусть $a_1 = u$, $b_1 = v$, $ak = 2b_{k-1} + a_{k-1}$, $b_k = 2a_{k-1} + b^2_{k-1}$, $k = 2, 3, \ldots$ Даны действительные u, v, натуральное n. Найти $\sum_{k=1}^{n}\dfrac{a_k b_k}{(k+1)!}$. Использовать рекурсивную подпрограмму.
	Вариант № 18.
1.	Даны вещественные массивы A(7), B(8), C(5). Составить программу с использованием подпрограммы для вычисления суммы и количества положительных и отрицательных элементов этих массивов.
2.	Даны действительные числа $a_1, a_2, \ldots, a_{20}$. Получить числа $b_1, b_2, \ldots, b_{20}$., где b_i – среднее арифметическое всех членов последовательности $a_1, a_2, \ldots, a_{20}$, кроме a_i ($i = 1, \ldots, 20$) .
3.	Пусть $v_1 = v_2 = 0$, $v_3 = 1{,}5$, $v_i = \dfrac{i+1}{i^3+1}\cdot v_{i-1} - v_{i-2}\cdot\cos v_{i-3}$, $i = 4, 5, \ldots$ Дано натуральное число n. Получить v_n. Использовать рекурсивную подпрограмму.
	Вариант 19
1.	Даны вещественные массивы A(10), B(15). Составить программу для вычисления значения $z=\dfrac{x+y}{kn}$, где x и k – сумма и количество положительных элементов массива A; y и n – сумма и количество отрицательных элементов массива B. Для вычисления суммы и количества положительных и отрицательных элементов массива использовать одну и ту же подпрограмму общего вида. Нулевых элементов в массивах нет.
2.	Дан двумерный вещественный массив A(8, 7). Найти сумму элементов в каждой строке матрицы между наибольшим ее элементом и наименьшим. Использовать подпрограмму.
3.	Пусть $x_0 = q$, $x_1 = \dfrac{1}{q}$, $x_k = \dfrac{x_{k-1}}{q} + q \cdot x_{k-2} - \dfrac{2}{q}$, $k = 2, 3, \ldots$ Дано действительное число q, натуральное n. Получить x_n. Использовать рекурсивную подпрограмму.
	Вариант № 20

1.	Даны действительные матрицы A (15, 10) и B (9, 7). Составить программу для нахождения сумм элементов каждой строки, используя подпрограмму-функцию.
2.	Даны действительные числа *s*, *t*. Получить *f*(*t*, -2*s*,1.17) + *f*(2.2, *t*, *s* – *t*), где $f(a,b,c)=\dfrac{2a-b-\sin c}{5+\lvert c\rvert}$.
3.	Пусть $a_0 = a_1 = 1$, $a_i = a_{i-2} + \dfrac{a_{i-1}}{2^{i-1}}$, i = 2, 3, ... Найти $\sum_{i=1}^{20} a_i$. Использовать рекурсивную подпрограмму.
	Вариант 21
1.	Даны действительные матрицы A (3, 10) и B (3, 7). Составить программу для нахождения сумм элементов каждого столбца матриц, используя подпрограмму.
2.	Дано действительное число x ($x \ge -1$). Составить программу для вычисления $z=\sum_{k=1}^{10}\dfrac{x^{2k+1}}{\left(\sqrt[2k+1]{x+1}\right)^2}$ используя функцию.
3.	Пусть $v_0 = v_1 = 1$, $v_i = \dfrac{-v_{i-1}}{\lvert+v_{i-1}\rvert+2}$ i = 2, 3, ... Дано натуральное n. Получить v_n. Использовать рекурсивную подпрограмму.
	Вариант 22
1.	Даны действительные массивы A (10) и B (7). Составить программу для вычисления значений функции $z(x,y)=\begin{cases} xa_{\max}, \text{если } x<y \\ yb_{\min}, \text{если } x\ge y \end{cases}$, где $x=\sum_{i=1}^{10} a_i$; $y=\sum_{i=1}^{7} b_i$; a_i и b_i – элементы массивов A и B. В одной подпрограмме вычислить сумму элементов массива, а в другой подпрограмме найти наибольший или наименьший элемент массива.
2.	Даны действительные числа *s*, *t*. Получить *g*(1.2, *s*) + *g*(*t*, *s*) – *g*(2*s* – 1, *st*), где $g(a,b)=\dfrac{a^2+b^2}{a^2+2ab+3b^2+4}$.
3.	Пусть $a_1 = b_1 = 1$, $a_k = \dfrac{1}{5}\left(\sqrt{b_{k-1}}+\dfrac{1}{5}\sqrt{a_{k-1}}\right)$, $b_k = a^2_{k-1} + b^2_{k-1}$, k = 2, 3, ... Дано натуральное число n. Найти $\sum_{k=1}^{n} a_k b_k$. Использовать рекурсивную подпрограмму.
	Вариант 23
1.	Даны действительные числа *x*, *y*. Составить программу для вычисления значений функции $z(x,y)=\dfrac{\sqrt[3]{5x^4+2x^3+x^2+9}}{12x^3-3x+70}+\sqrt[3]{12y^7+y^4+9y^2+11}$. Вычисление многочлена осуществлять в подпрограмме по схеме Горнера.
2.	Дано действительное число y. Получить $\dfrac{1.7t(0.25)+2t(1+y)}{6-t(y^2-1)}$, где $t(x)=\sum_{k=0}^{10}\dfrac{x^{2k+1}}{(2k+1)!}$.
3.	Пусть $x_1 = y_1 = 1$, $x_i = 0{,}3 \cdot x_{i-1}$, $y_i = x_{i-1} + y_{i-1}$, i = 2, 3, ... Дано натуральное число

	n. Найти $\sum_{i=1}^{n}\frac{x_i}{1+\lvert y_i\rvert}$. Использовать рекурсивную подпрограмму.
	Вариант 24
1.	Даны натуральные числа a, b, c, d, e. Составить программу для нахождения наибольшего общего делителя пяти целых положительных чисел. Наибольший общий делитель двух чисел вычислять в подпрограмме.
2.	Даны действительные числа a, b, c. Получить $\frac{\max(a, a+b)+\max(a, b+c)}{1+\max(a+bc, 1, 15)}$.
3.	Пусть $a_1 = b_1 = 1$, $a_k = 3b_{k-1} - 2a_{k-1}$, $b_k = 2a^2_{k-1} + b_{k-1}$, $k = 2, 3, \ldots$ Дано натуральное n. Найти $\sum_{k=1}^{n}\frac{2^k}{\left(1+a_k^2+b_k^2\right)\cdot k}$. Использовать рекурсивную подпрограмму.
	Вариант 25
1.	Даны натуральные числа n, m, k, l, действительные двумерные массивы A(n, m), B (k, l). Составить программу для вычисления сумм положительных элементов каждой строки матриц, если значения n, m, k, l не превосходят 20. Результаты запомнить в массивах S и R. Использовать подпрограмму.
2.	Дана матрица A(8, 7). Найти сумму элементов в каждом столбце между наибольшим его элементом и наименьшим. Использовать подпрограмму.
3.	Пусть $x_0 = 1$, $x_k = \frac{2-x_{k-1}^3}{5}+\frac{1}{x_{k-1}}$, $k = 1, 2, \ldots$ Найти первый член x_n, для которого $\lvert x_n - x_{n-1}\rvert < 10^{-5}$. Использовать рекурсивную подпрограмму.
	Вариант 26
1.	Даны массивы T(10) и P(10). Составить программу для вычисления значения $D=\sum_{k=1}^{10}(t_k - p_k)^2$.
2.	Даны действительные числа a, b. Получить $u = \min(a, b)$, $v = \min(ab, a + b)$, $\min(u + v^2, 3.1415926)$.
3.	Даны положительные действительные числа a, x, ε . В последовательности y_1, y_2, ... образованной по закону $y_0 = a$, $y_i = \frac{1}{2}\left(y_{i-1}+\frac{x}{y_{i-1}}\right)$, $i = 1, 2, \ldots$, найти первый член y_n, для которого выполнено неравенство $\lvert y^2_n - y^2_{n-1}\rvert < \varepsilon$. Использовать рекурсивную подпрограмму.
	Вариант 27
1.	Дан двумерный массив A(10, 15). Упорядочить по возрастанию элементы строки матрицы. Упорядочивание элементов по возрастанию осуществить в подпрограмме.
2.	Даны действительные числа s, t. Получить $h(s, t) + \max(h^2(s, t), h^4(\text{s} - \text{t}, \text{st}) + \text{h}(1, 1)$, где $h(a,b)=\frac{a}{1+b^2}+\frac{b}{1+a^2}-(a-b)^3$.
3.	Пусть $x_1 = x_2 = x_3 = 1$, $x_i = x_{i-1} + x_{i-3}$, $i = 4, 5, \ldots$ Найти $\sum_{i=1}^{10}\frac{x_i}{2^i}$. Использовать рекурсивную подпрограмму.
	Вариант 28
1.	Составить программу для вычисления функции $z=\frac{(a-x_1)(a-x_2)}{(b-y_1)(b-y_2)}$, где x_1, x_2 – корни уравнения $ax^2 - x + 1 = 0$; y_1, y_2 – корни уравнения $2y^2 + y - b = 0$. Если

	дискриминант квадратного уравнения меньше нуля (то есть корни мнимые) то считать их равными нулю. Для вычисления корней использовать подпрограмму.
2.	Даны действительные числа x, a_0, a_1, a_2, ..., a_6. Получить значение выражения $L(x) = p(x + 1) – p(x)$ где $p(y) = a_6y^6 + a_5y^5 + ... + a_0$.
3.	Пусть $a_0 = 1$, $a_k = k \cdot a_{k-1} + \frac{1}{k}$, $k = 1, 2, ...$ Дано натуральное число n. Получить a_n. Использовать рекурсивную подпрограмму.
	Вариант 29
1.	Составить программу для зачисления на стипендию студентов групп А, В, С, состоящих соответственно из 20, 24 и 23 человек, используя подпрограмму общего вида. Зачисление на стипендию осуществлять, если в оценках студента за экзамены не более половины «удовлетворительно» и нет оценок «неудовлетворительно».
2.	Даны действительные числа s, t, a_0, ..., a_{12}. Получить $p(1) – p(t) + p^2(s – t) – p^3(1)$, где $p(x) = a_{12}x^{12} + a_{11}x^{11} + ... + a_0$.
3.	Пусть $a_1 = u$, $b_1 = v$, $a_k = \frac{2b_{k-1}}{a_{k-1}}$, $b_k = \frac{2a_{k-1}^2}{b_{k-1}^3}$, $k = 2, 3, ...$ Даны действительные числа u, v, натуральное n. Найти $\sum_{k=1}^{n} \frac{a_k b_k}{(k+1)!}$. Использовать рекурсивную подпрограмму.
	Вариант 30
1.	Составить программу для определения мест, занятых гимнастами на соревнованиях, используя подпрограмму общего вида. Соревнования проводились в три потока – юниоры, перворазрядники, мастера спорта. Результаты, показанные гимнастами на отдельных снарядах, для каждого потока сведены в матрицы А (20, 6), В (22, 6), С (25, 6).
2.	Даны целые числа n_0, d_0, n_1, d_1, ..., n_7, d_7, (a, b, d_0, ..., d_7, a, $b \neq 0$), вычислить по схеме Горнера значение многочлена $G(x) = \frac{n_7}{d_7}(x)^7 + \frac{n_6}{d_6}(x)^6 + \ldots + \frac{n_0}{d_0}$ в точке $x_0 = \frac{a}{b}$.
3.	Пусть $v_1 = v_2 = 1$, $v_3 = 0{,}5$, $v_i = \frac{i-2}{-i^3+1} \cdot v_{i-1} - v_{i-3}$, $i = 4, 5, ...$ Дано натуральное n. Получить v_n. Использовать рекурсивную подпрограмму.

Решение типового варианта

Рассмотрим пример оформления работы.

Пусть вариант студента 31.

	Вариант 31
1.	Даны целочисленные матрицы A (5, 6), B (6, 6), C (4, 6). Составить программу для заполнения одномерных массивов AS(5), BS(6), CS(4) суммами элементов в каждой строке, используя подпрограмму общего вида.
2.	Даны натуральные числа x, a_1, a_2, ..., a_{12}, вычислить по схеме Горнера значение многочлена $G(x)=\left(\frac{x}{a_{12}}\right)^{12}+\left(\frac{x}{a_{11}}\right)^{11}+\ldots+\frac{1}{a_1}$, используя подпрограмму общего вида.
3.	Пусть $v_1 = v_2 = 1{,}5$, $v_3 = 2$, $v_i=\frac{v_{i-1}}{i^{-1}+10}+v_{i-2}-v_{i-3}$, $i = 4, 5, \ldots$ Дано натуральное n. Получить v_n. Использовать рекурсивную подпрограмму.

Сформируем отчет к лабораторной работе 7.

1. Титульный лист.

2. Сформируем задание варианта 31.

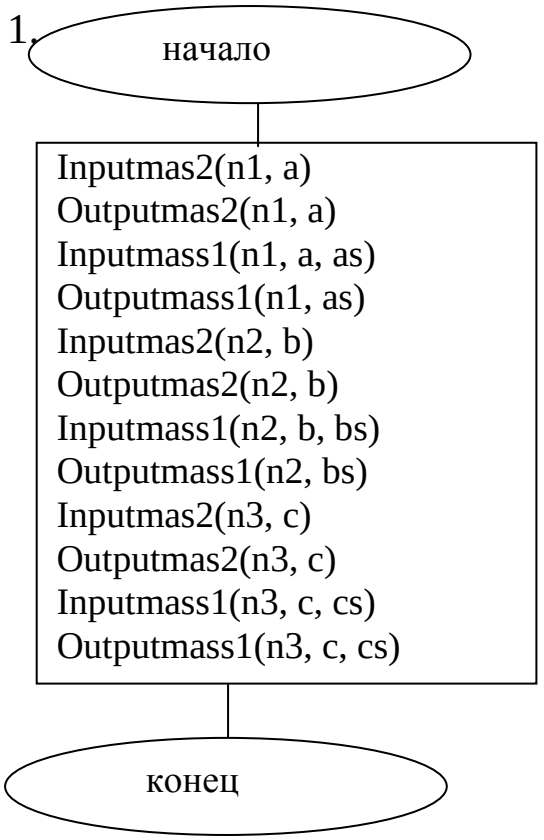
1. Даны целочисленные матрицы A (5, 6), B (6, 6), C (4, 6). Составить программу для заполнения одномерных массивов AS(5), BS(6), CS(4) суммами элементов в каждой строке, используя подпрограмму общего вида.

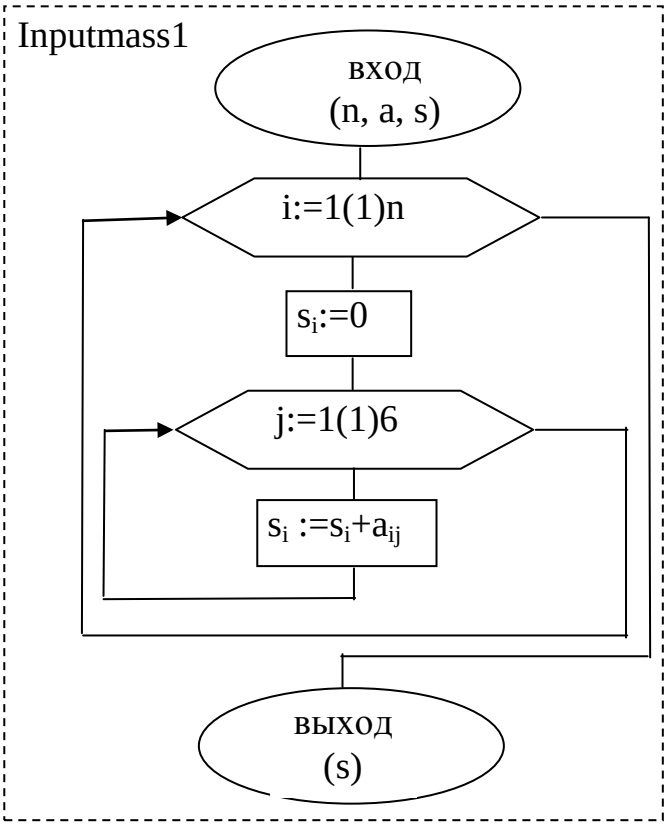
2. Даны целые числа x, a_1, a_2, ..., a_{12}. Вычислить по схеме Горнера значение многочлена $G(x)=\left(\frac{x}{a_{12}}\right)^{12}+\left(\frac{x}{a_{11}}\right)^{11}+\ldots+\frac{1}{a_1}$, используя подпрограмму общего вида.

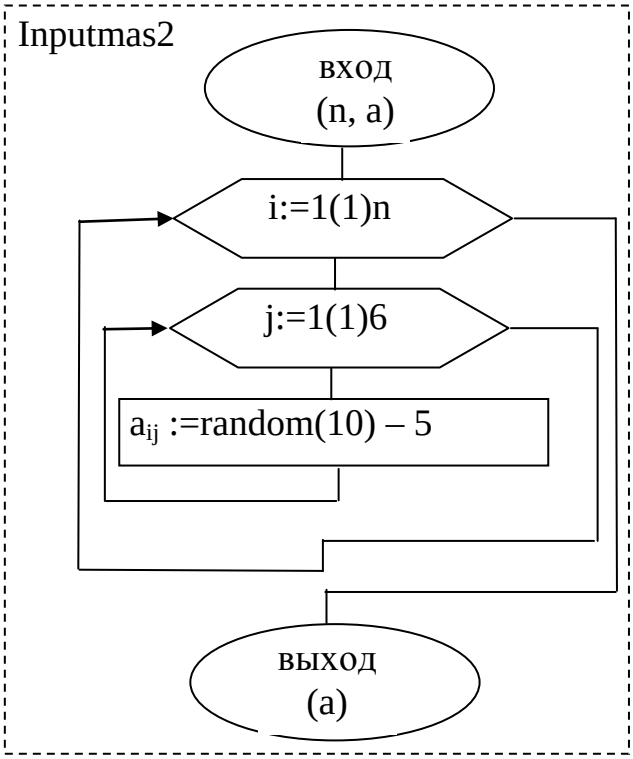
3. Пусть $v_1 = v_2 = 1{,}5$, $v_3 = 2$, $v_i=\frac{v_{i-1}}{i^{-1}+10}+v_{i-2}-v_{i-3}$, $i = 4, 5, \ldots$ Дано натуральное n. Получить v_n. Использовать рекурсивную подпрограмму.

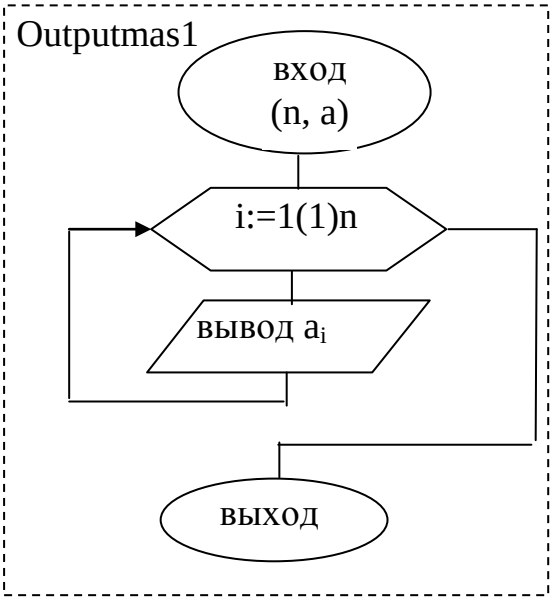
4. Составим блок-схему алгоритмов заданий варианта.

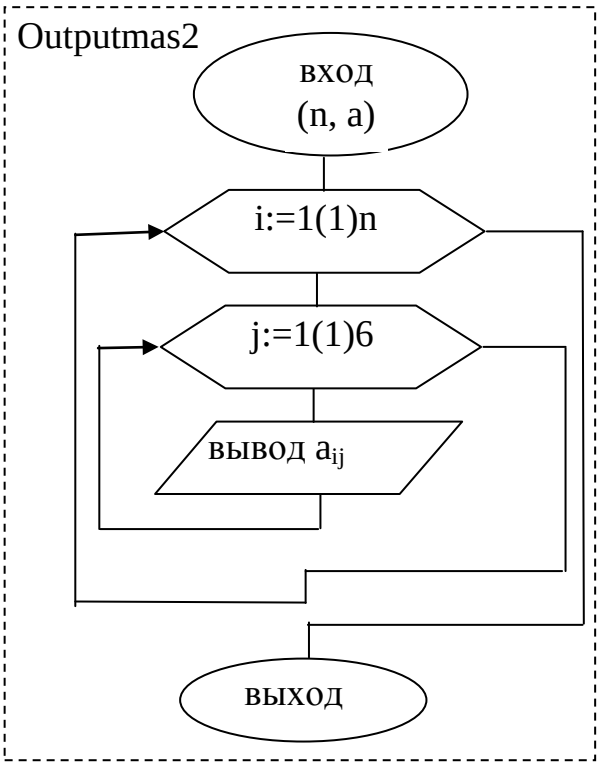
1.


начало
Inputmas2(n1, a)
Outputmas2(n1, a)
Inputmass1(n1, a, as)
Outputmass1(n1, as)
Inputmas2(n2, b)
Outputmas2(n2, b)
Inputmass1(n2, b, bs)
Outputmass1(n2, bs)
Inputmas2(n3, c)
Outputmas2(n3, c)
Inputmass1(n3, c, cs)
Outputmass1(n3, c, cs)
конец

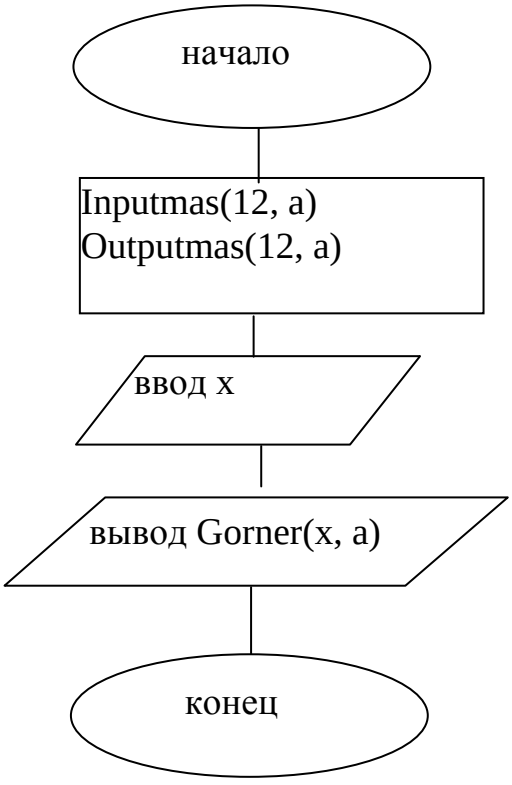

Inputmass1
вход
(n, a, s)
i:=1(1)n
s_i:=0
j:=1(1)6
s_i :=s_i+a_{ij}
выход
(s)

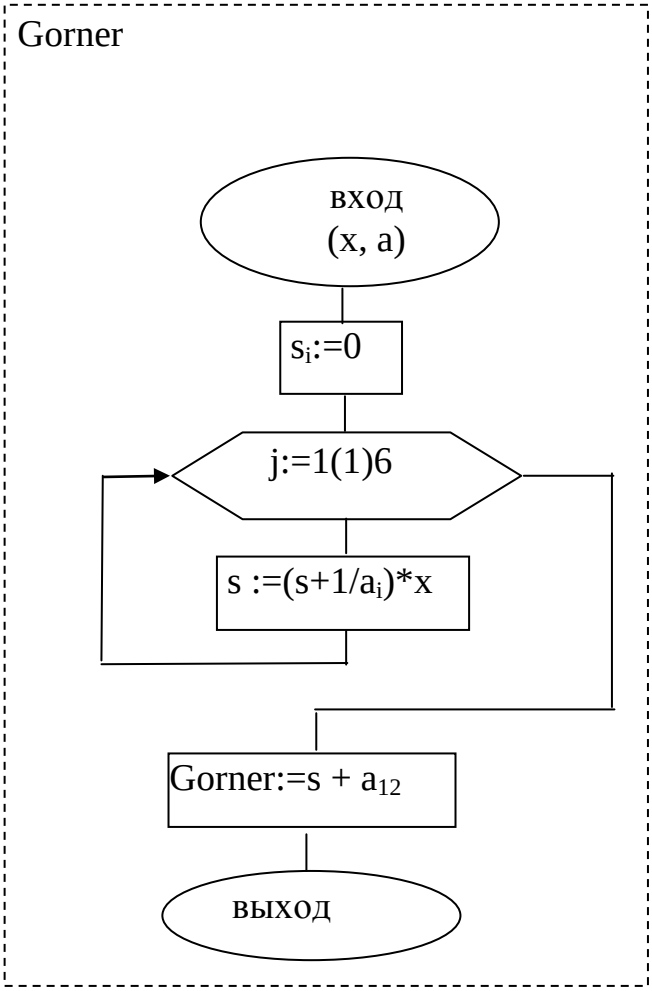

Inputmas2
вход
(n, a)
i:=1(1)n
j:=1(1)6
a_{ij} :=random(10) – 5
выход
(a)

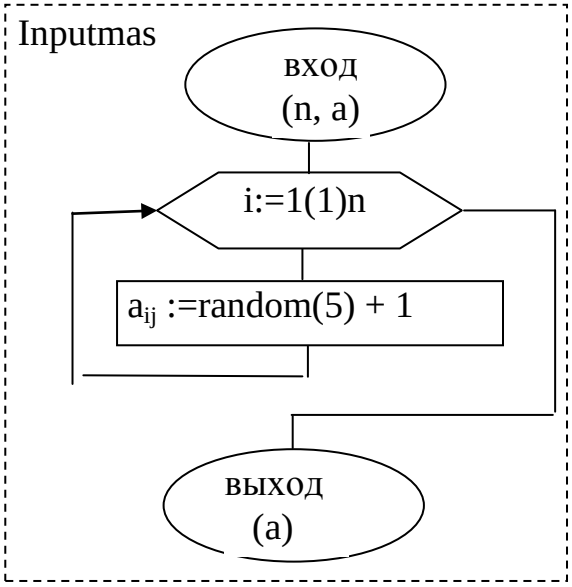

Outputmas1
вход
(n, a)
i:=1(1)n
вывод a_i
выход

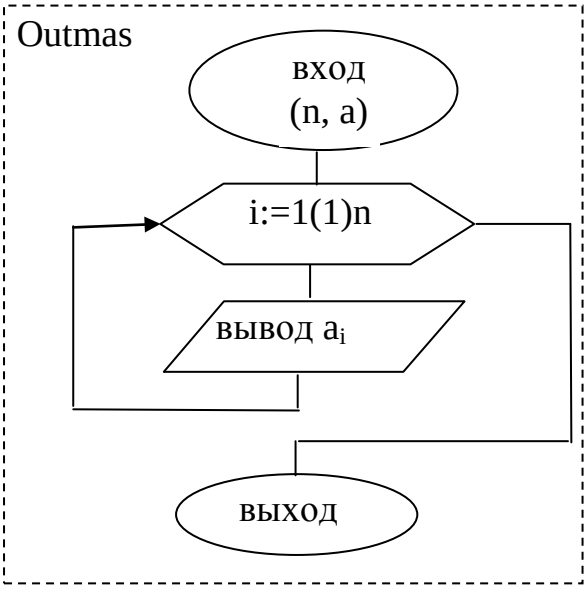

Outputmas2
вход
(n, a)
i:=1(1)n
j:=1(1)6
вывод a_{ij}
выход

2.

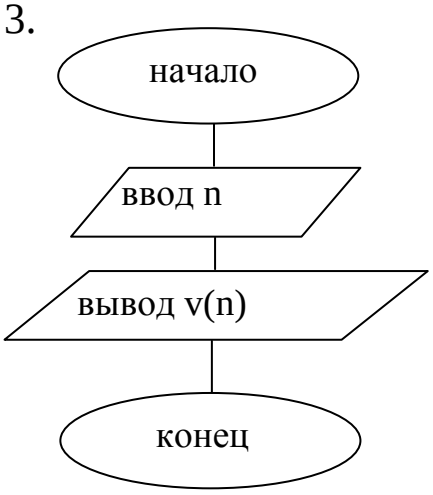

начало
Inputmas(12, a)
Outputmas(12, a)
ввод x
вывод Gorner(x, a)
конец

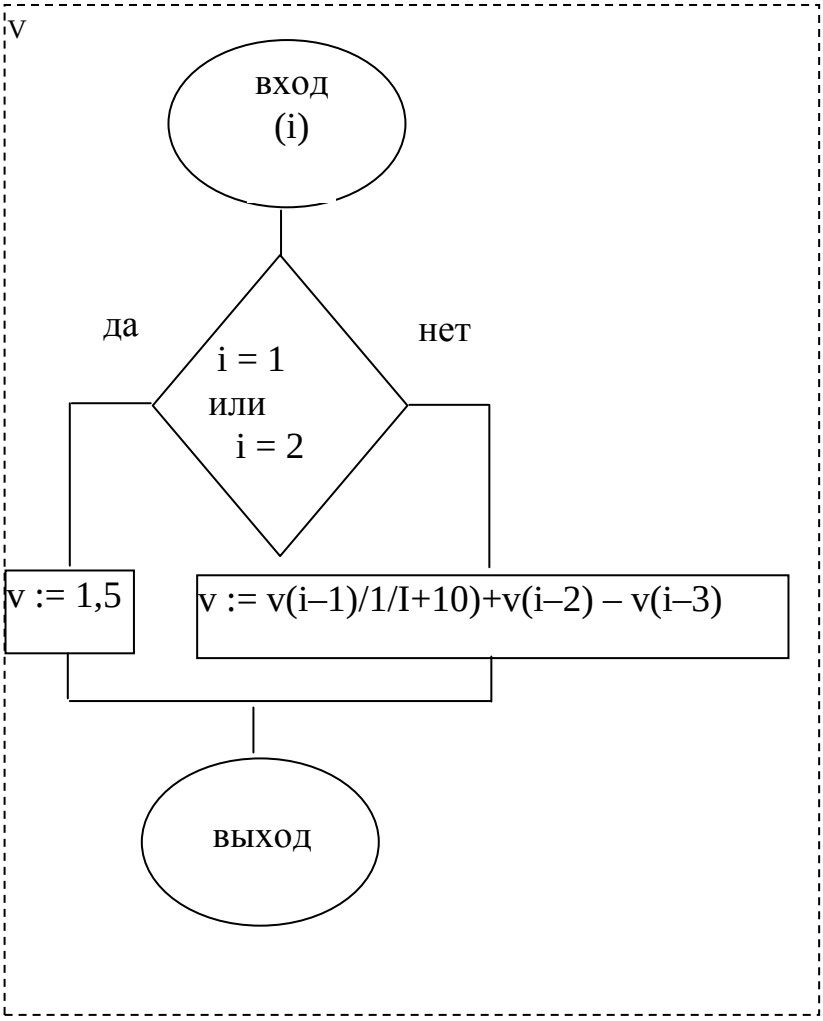

Gorner
вход
(x, a)
s_i:=0
j:=1(1)6
s :=(s+1/a_i)*x
Gorner:=s + a_{12}
выход





3.





5. Напишем программы.

```
(*1.*)
uses crt;
const n1=5;n2=6;n3=4;
type mas1=array[1..n2] of integer;
```

```
     mas2=array[1..n2,1..6] of integer;
const
  a:mas2=(( 1, 2,-2, 2,-2, 0),
          ( 1,-4,-5, 2, 3, 4),
          ( 0, 1,-5, 4, 0, 3),
          (-5,-2, 2,-4, 3, 3),
          (-5,-2, 4,-1, 0, 0),
          ( 0, 0, 0, 0, 0, 0));
  b:mas2=((-4,-2,-2, 4, 1,-4),
          ( 3, 0, 0,-4,-3,-1),
          (-3,-5, 2,-2, 3, 4),
          (-5, 3,-5,-2, 4, 3),
          (-2, 4, 4, 2,-3, 4),
          ( 2,-4,-2,-1,-2, 2));
  c:mas2=((-1, 0, 0, 2,-4, 1),
          ( 4, 3,-5, 2, 3, 1),
          (-3, 4, 2,-2,-5, 3),
          (-1,-3,-1, 4, 0,-1),
          ( 0, 0, 0, 0, 0, 0),
          ( 0, 0, 0, 0, 0, 0));
var
{a,b,c:mas2}
as,bs,cs:mas1;
i,j,na,nb,nc:byte;
procedure Inputmas2(n:byte; var a:mas2);
var i,j:byte;
begin
for i:=1 to n do
    for j:=1 to 6 do
        a[i,j]:=random(10)-5;
end;
procedure Outputmas2(n:byte; var a:mas2);
var i,j:byte;
```

```
begin
writeln;
for i:=1 to n do begin
    for j:=1 to 6 do
        write(a[i,j]:6);
    writeln;
end;
    writeln;
end;
procedure Inputmass1(n:byte;a:mas2; var s:mas1);
var i,j:byte;
begin
for i:=1 to n do begin
    s[i]:=0;
    for j:=1 to 6 do
        s[i]:=s[i]+a[i,j]
end;
end;
procedure Outputmas1(n:byte; a:mas1);
var i:byte;
begin
for i:=1 to n do begin
    write(a[i]:6);
end;
    writeln;
    readkey
end;
begin
clrscr;
randomize;
{Inputmas2(n1, a);}
Outputmas2(n1, a);
Inputmass1(n1, a, as);
```

```
Outputmas1(n1, as);
{Inputmas2(n2, b);}
Outputmas2(n2, b);
Inputmass1(n2, b, bs);
Outputmas1(n2, bs);
{Inputmas2(n3, c);}
Outputmas2(n3, c);
Inputmass1(n3, c, cs);
Outputmas1(n3,cs);
end.
(* 2.*)
uses crt;
type mas=array[1..12] of shortint;
const
{   a:mas=(3,3,5,5,3,5,2,3,3,3,5,1);}
{   a:mas=(1,2,5,3,1,5,3,5,5,2,3,2);}
   a:mas=(4,3,3,3,5,1,3,4,2,2,2,3);
var {a:mas;}
i,x:byte;
function Gorner(x:real;a:mas):real;
var s, p:real;
Begin
s:=0;
for i:=1 to 11 do begin
    s:=(s+1/a[i])*x
end;
Gorner:=s+a[12]
End;
procedure Inputmas(n:byte; var a:mas);
var i:byte;
begin
for i:=1 to n do
  a[i]:=random(5)+1;
```

```
end;
procedure Outputmas(n:byte; var a:mas);
var i:byte;
begin
    for i:=1 to n do
        write(a[i]:6);
end;
begin
   clrscr;
   randomize;
{   Inputmas(12, a);}
   Outputmas(12, a);
   writeln;
   writeln('WWod x');
   readln(x);
   writeln(Gorner(x,a):6:3);
   readkey
end.
(* 3.*)
uses crt;
var n:integer;
function v(i:integer):real;
Begin
if (i=1) or (i=2) then v:=1.5
else if (i=3) then v:=2
else v:=v(i-1)/(1/i+10)+v(i-2)-v(i-3)
end;
begin
    writeln('Wwod n');
    readln(n);
    writeln(v(n):6:3);
    readkey
end.
```

*)

6. Представленные программы не содержат ошибок, подберем значения для тестирования программ.

1. a=((1, 2,-2, 2,-2, 0),
   (1,-4,-5, 2, 3, 4),
   (0, 1,-5, 4, 0, 3),
   (-5,-2, 2,-4, 3, 3),
   (-5,-2, 4,-1, 0, 0),
   (0, 0, 0, 0, 0, 0)); AS = (1, 1, 3, –3 , –4)
   b=((-4,-2,-2, 4, 1,-4),
   (3, 0, 0,-4,-3,-1),
   (-3,-5, 2,-2, 3, 4),
   (-5, 3,-5,-2, 4, 3),
   (-2, 4, 4, 2,-3, 4),
   (2,-4,-2,-1,-2, 2)); BS = (–7, –5, –1, –2, 9, –5)
   c=((-1, 0, 0, 2,-4, 1),
   (4, 3,-5, 2, 3, 1),
   (-3, 4, 2,-2,-5, 3),
   (-1,-3,-1, 4, 0,-1),
   (0, 0, 0, 0, 0, 0),
   (0, 0, 0, 0, 0, 0)); CS = (–2, 8, –1, –2)
2. а) a=(3,3,5,5,3,5,2,3,3,3,5,1) x = 0 g(0) = 1
   б) a=(1,2,5,3,1,5,3,5,5,2,3,2) x = 1 g(1) = 6,8
   в) a=(4,3,3,3,5,1,3,4,2,2,2,3) x = 2 g(2) = 1223,6
3. а) n = 2 $v_2 = 1,5$
   б) n = 5 $v_5 = 0,519$
   в) n = 14 $v_{14} = –6,716$

7. В текстовом редакторе WORD оформим отчет в соответствии с содержанием.

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1 .Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Абрамов, С. А. Начала информатики / С. А. Абрамов, Е. В. Зима. – М. : Наука, 1989. – 256 с.

3. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

4. Довгаль, С. И. Персональные ЭВМ : Турбо Паскаль V 7.0. Объектное программирование. Локальные сети : учебное пособие / С. И. Довгаль, Б. Ю. Литвинов, А. И. Сбитнев.

5. Новичков, В. С. Паскаль : учеб. пособие для сред. спец. учеб. заведений / В. С. Новичков, Н. И. Парфилова, А. Н. Пылькин. – М. : Высш. шк., 1990. – 223 с. : ил. – (Алгоритмические языки в техникуме).

6. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

Вопросы для самопроверки

1. Сформулируйте в общем виде случаи, требующие подпрограмм, перечислите виды подпрограмм. Укажите служебные слова, задающие каждый из этих видов. Поясните различия в написании заголовков каждого из этих видов.

2. Сформулируйте, как организуется обращение к подпрограммам вида «процедура» и вида «функция».

3. Дайте определение рекурсивных подпрограмм. Приведите пример рекурсивной подпрограммы. Перечислите виды рекурсивных подпрограмм

4. Сформулируйте основные отличия локальных переменных от глобальных.

5. Сформулируйте определение формальных параметров. В чем отличие параметров-переменных от параметров-значений. Укажите служебные слова и структуры, в которых они используются.

6. Перечислите основные отличия процедур от функций.

7. Дайте определение косвенной рекурсии. Проиллюстрируйте косвенную рекурсию на примере подпрограмм с именами A и B

8. Оформите процедурой определение наибольшего числа из трех чисел.

9. Оформите функцией определение наибольшего числа из трех чисел.

10. Запишите в виде процедуры определение максимального числа в числовом массиве *A*(10).

11. Запишите в виде функции определение максимального числа в числовом массиве *A*(10).

12. Запишите программу проверки строки на палиндром. Проверку оформить процедурой.

13. Запишите программу проверки строки на палиндром. Проверку оформить функцией.

14. Оформите в виде процедуры задачу поиска номера максимального числа в массиве *A*(10).

15. Оформите в виде процедуры задачу поиска номера минимального числа в массиве *A*(10).

16. Оформите в виде процедуры задачу «Определение среднего арифметического элементов массива *A*(10)».

17. Оформите в виде функции задачу «Определение среднего арифметического элементов массива *A*(10)».

18. Заданы два символьных массива одинакового размера *A*(10), *B*(10). Проверьте их на равенство друг другу. Проверку оформить процедурой.

19. Определите, все ли правильно составлено в следующем описании процедуры без параметров, и если нет, то что и почему неправильно: **PROCEDURE DIS; D:=SQR(B)-4*A*C;**

20. Определите, все ли правильно составлено в следующем описании процедуры без параметров, и если нет, то что и почему неправильно: **PROCEDURE DIS; VAR D: REAL; D:= SQR(B) –4*A* C;**

21. Определите, все ли правильно составлено в следующем описании процедуры без параметров, и если нет, то что и почему не правильно: **PROCEDURE DIS; BEGIN D:= SQR(B) –4*A*C END;**

22. Рассмотрим описание процедуры P без параметров **PROCEDURE P; BEGIN X:= X – Y; Y:= X + Y END;** Пропишите последствия обращения к процедуре P, если до обращения переменная X приобрела значение целого числа 5, а Y – значение целого числа –7.

23. Рассмотрим описание процедуры Q без параметров **PROCEDURE Q; BEGIN Y:= X + Y; X:= X – Y END;** Пропишите последствия обращения к процедуре Q, если до обращения переменная X приобрела значение целого числа 5, а Y – значение целого числа –7.

24. Если в программе и в процедуре описана одна и та же переменная, то какое из этих описаний имеет силу в процедуре?

25. Процедура описана следующим образом: **PROCEDURE F(VAR X, Y: INTEGER) BEGIN X:= У END;** Допустимо ли обращение к процедуре, имеющее вид f (a, b – 1)?

26. Предположим, что функции abs и odd не являются стандартными. Составьте их описания.

27. Составьте функцию, определяющую наименьшее из двух чисел.

ЛАБОРАТОРНАЯ РАБОТА 8. СИМВОЛЬНЫЕ ДАННЫЕ В ТУРБО ПАСКАЛЬ. ТИП СИМВОЛ И ТИП СТРОКА, ПРОЦЕДУРЫ И ФУНКЦИИ РАБОТЫ С СИМВОЛЬНЫМИ ДАННЫМИ (4 ЧАСА)

Цель работы: приобретение навыков работы с символьными данными в интегрированной среде Турбо Паскаль. Знание особенности использования данных типов CHAR и STRING.

Теоретические положения

Символьные данные в Турбо Паскаль

Мы уже познакомились со структурированным типом данных МАССИВ. Напомним, что есть еще три структурированных типа: ЗАПИСИ, МНОЖЕСТВА и ФАЙЛЫ. Все они характеризуется множественностью образующих. Еще один тип, его в некоторых методических изданиях называют полуструктурированным типом. Он отсутствует в стандартном Паскале, его концепция обработки символьных данных заключалась в широком использовании типа CHAR, как единичных символов, так и цепочки. Но по мере популяризации языка в среде профессиональных разработчиков программ необходимость повысить удобство работы с символьными данными потребовала введения особого типа СТРОКА (STRING). Он не относится к простым типам данных и занимает промежуточное место между простыми и структурированными типами данных.

Символьный тип

Символьный тип CHAR представляет собой тип данных, предназначенный для хранения одного символа (буквы, знака или кода). В переменную этого типа на компьютере IBM PC может быть помещен любой из 256 символов расширенного кода ASCII. Это буквы [‘A’..’Z,’a’..’z’], цифры [‘0’..’9’], знаки препинания и специальные символы. Переменная типа CHAR в памяти занимает один байт так же, как и переменная типа BYTE. Этот тип данных в ТП 7.0 обладает некоторыми особенностями. Проиллюстрируем их на следующем примере:

```
uses crt;
var
  ch: char;
begin
  clrscr;
  ch:='A'; {переменной ch присваивается символ 'A'}
  writeln(ch); {вывод на экран символа 'A'}
  writeln(ord(ch)); {функция ord(ch) возвращает ASCII-код литеры 'A'}
  ch := #65; {переменной ch присваивается символ с ASCII кодом 65,
              то есть символ 'A'}
```

```
  writeln(ch); {вывод на экран символа 'A'}
  readkey
end.
```

Обычно значения для переменных типа CHAR задаются так же, как и строки символов, – в апострофах. Кроме того, имеется возможность задавать значения указанием непосредственно числового значения ASCII-кода. В этом случае Вы должны перед числом, обозначающим код ASCII символа, поставить знак (#), как это показано в примере. Можно также указывать в качестве значений переменных типа CHAR специальные символы, часто называемые «управляющими кодами». Их указывают с помощью значка ^ и последующей литеры. Управляющим кодам соответствуют вполне определенные значения ASCII-кода, и это дает возможность, например, вместо ^a вводить #1, так как код Ctrl-A равен 1. Управляющие коды часто скрытно используются встроенными процедурами WriteLn для форматного вывода (перевода курсора на новую строку). Кроме того, программист может сам явно включать управляющие коды в поток выводимой информации, управляя ее расположением на экране (как это показано в нашем примере), а также создавать «звуковое сопровождение», которое будет предварять процесс вывода информации.

Операции над символами

В Турбо Паскале символы, то есть переменные типа CHAR, можно лишь присваивать и сравнивать друг с другом. При сравнении символов фактически сравниваются не они, а их ASCII-коды, при этом один символ считается больше другого только в том случае, если он имеет больший ASCII-код. Например:

'B'>'A' (ASCII-код символа 'A'=65, символа 'B'=66)

'A'<'a' (ASCII-код символа 'a'=97)

'A'='A'

В таблице 1 приведены функции, которые могут применяться к переменным с типом данных CHAR.

Таблица 1

Функция	Назначение
Chr(X : BYTE) : CHAR	Возвращает символ, соответствующий ASCII-коду числа X

Ord(X : CHAR) : BYTE	Возвращает число, соответствующее символу X в ASCII-таблице
UpCase(X : CHAR) : CHAR	Преобразует символы из строчных букв в прописные
Pred(X: CHAR) : CHAR	Возвращает символ, который предшествует в ASCII-таблице символу X
Succ(X : CHAR) : CHAR	Возвращает символ, который следует в ASCII-таблице за символом X

Упорядоченность кодов ASCII, соответствующих символам литер, позволяет использовать при их обработке операции сравнения (например для сортировки) подобно тому, как это делается для целых и действительных чисел. Все коды литер латинского алфавита расположены в одной части ASCII-таблицы, тогда как коды литер русского алфавита – в разных ее частях, хотя в Турбо Паскале упорядочены они одинаково, то есть алфавитному порядку следования литер соответствует упорядоченная по возрастанию последовательность их кодов. Сортировка русских слов по алфавиту программируется сложнее, чем английских.

Объявление строчных типов и строчных переменных

Переменные типа STRING могут быть объявлены следующим образом:

```
VAR
    Ch_l: STRING;
    Ch_2: STRING[255];
```

Переменная типа STRING объявляется, как правило, путем указания имени переменной, зарезервированного слова STRING и указания (в квадратных скобках) максимального размера (длины) строки, которая может храниться в этой переменной. Если максимальный размер строки не указан, то он автоматически принимается равным 255 – максимально возможная длина строки.

Операции со строками

В Турбо Паскале всегда была преемственность версий, то есть программа, созданная в версии 1.0 с минимальными модификациями

работает в любой следующей вплоть до 7.0, поэтому существуют два пути обработки переменных типа STRING. Первый путь предполагает обработку всей строки как единого целого, то есть единого объекта. Второй путь рассматривает строку как составной объект, состоящий из отдельных символов, то есть элементов типа CHAR, которые при обработке доступны каждый в отдельности. Так, первый путь предоставляет возможность присвоения строчной переменной за одну операцию значения целой строки символов:

Ch_str :=‘ Это – строка !’ ;

Присваиваемое значение строки, так же как и отдельный символ типа CHAR, заключается в апострофы. Если апострофы опущены, то компилятор рассматривает приведенный фрагмент текста как числовую величину или как идентификатор.

Турбо Паскаль позволяет выполнять операции объединения (сцепления) нескольких строк в процессе их присваивания какой-либо переменной: Ch_str := ‘Это’ + ‘-‘ + ‘ строка ’+’ !’;

В результате такой операции в переменной Ch_str будет то же самое содержимое, что и в предыдущем примере.

Второй подход обеспечивает доступ к отдельным символам строки по номеру их позиции:

```
Ch_str [1] := ‘A’ ;      {Занести в качестве первого символа строки ‘A’}
Ch__str[5] := #49;       {Занести в качестве пятого символа строки ‘1’
                          (ASCII-код ‘1’ соответствует числу 49)}
```

Для доступа к отдельному символу в строке необходимо указать имя строки и в квадратных скобках номер позиции элемента (символа) в строке. При этом по отношению к отдельному символу строки возможны все те же операции, что и к переменной типа CHAR. В частности, возможны взаимные операции присвоения значений.

Так как в рамках Турбо Паскаля 7.0 с данными типа STRING связан целый набор операций, целесообразно остановиться на нем более подробно.

Переменная типа STRING состоит из цепочки символов, то есть элементов типа CHAR. Строки могут выводиться на экран монитора посредством стандартных процедур Write и WriteLn и вводится с помощью стандартной процедуры ReadLn или Read. В большинстве случаев переменные типа STRING используются для хранения слов и сообщений, состоящих из нескольких символов (см. табл. 2-4).

Таблица 2

Символьные процедуры

ПРОЦЕДУРЫ РАБОТЫ СО СТРОКАМИ	
ИМЯ	**НАЗНАЧЕНИЕ**
Delete	Удаление подстроки из строки
Insert	Помещение подстроки в строку
Str	Преобразование числа в строковую переменную
Val	Преобразование символьного представления числа в численное представление

Таблица 3

ФУНКЦИИ РАБОТЫ СО СТРОКАМИ	
ИМЯ	**НАЗНАЧЕНИЕ**
Concat	Объединение строк
Copy	Выделение подстроки
Length	Длина строки
Pos	Поиск подстроки в строке

Таблица 4

ФУНКЦИИ ПРЕОБРАЗОВАНИЯ ТИПОВ	
ИМЯ	**НАЗНАЧЕНИЕ**
Chr	Получение символа по его коду
Ord	Порядковый номер величины перечисляемого типа

Процедура Delete

procedure Delete (var S: string; Index: Integer; Count: Integer);

Удаляет подстроку из строки S. Для добавления подстроки в строку следует использовать процедуру Insert.

S – исходная строка;

Index – номер первого удаляемого символа (если номер больше размера строки, символы не удаляются);
Count – число удаляемых символов (если символов в строке недостаточно, удаляется остаток символов).

Процедура Insert

procedure Insert(Source: string; var S: string; Index: Integer);

Помещает подстроку Source в строку S (если получается строка слишком большого размера, то она усекается до 255 символов). Удаляется подстрока из строки процедурой Delete. Для подсоединения подстроки к началу или концу исходной строки можно использовать также функцию Concat или операцию конкатенации (+).
S – исходная строка;
Source – подстрока, помещаемая в строку;
Index – номер позиции исходной строки, начиная с которой помещается подстрока.

Процедура Str

procedure Str(X [: M [: N]]; var S: <строковый тип>);
Преобразует число в последовательность символов. Преобразование символьного представления числа в двоичное осуществляется процедурой Val. X – выражение вещественного или целого типа;
S – строка типа string, в которую записывается символьное представление числа.
M, N – форматы вывода целого типа;

Процедура Val

procedure Val(S: <строковый тип>; var V; var Code: Integer);
Преобразует символьное представление числа в численное представление. Преобразование числа в его символьное представление осуществляется процедурой Str.
S – строка типа string с символьным представлением числа;
V – переменная целого или вещественного типа для записи двоичного представления числа;

Code – номер неправильного символа (0 – если изображение числа правильное).

Функция Concat

function Concat(Sl [, S2, ..., Sn]: string): string;

Объединяет несколько строк в одну (при необходимости усекает чрезмерно большую строку до 255 символов). Для помещения подстроки в любое место строки используется процедура Insert.

S1, S2, ..., Sn – объединяемые строки.

Функция Copy

function Copy(S: string; Index: Integer; Count: Integer): string;

Создает подстроку строки S.

S – исходная строка;

Index – номер первого выделяемого символа строки (если значение больше размера строки, возвращается пустая строка);

Count – число выделяемых символов (если всех необходимых символов в строке нет, возвращается имеющийся остаток строки).

Функция Length

function Length(S: string): Integer;

Возвращает текущий размер строки.

S – строка, у которой определяется размер.

Функция Pos

function Pos(Substr,S: string): Byte;

Поиск последовательности Substr в строке S (результат равен номеру первого символа строки S, с которого начинается искомая последовательность, или 0, если такой последовательности в строке нет.

Substr – искомая последовательность;

S – строка, в которой ищется последовательность.

Функция Chr

function Chr(X: Byte): Char;

Возвращает символ с указанным кодом. Получить код символа можно с помощью функции Ord.

X – число, определяющее код символа.

Функция Ord

function Ord(x: <порядковый тип>): Longint;

Возвращает порядковый номер значения выражения порядкового типа (нумерация начинается с нуля).

X – выражение любого порядкового типа.

Некоторые коды ASCII

♦ от 48 до 57 – коды цифр от 0 до 9;

♦ от 65 до 90 – прописные буквы латиницы (A – 65, B – 66 и т. д.);

♦ от 97 до 122 – строчные буквы латиницы (a – 97, b – 98 и т. д.);

♦ от 128 до 159 – прописные буквы кириллицы (А – 128, Б – 129, и т. д.);

♦ от 160 до 175 и от 224 до 239 – строчные буквы кириллицы;

♦ клавиша Tab – 9, клавиша Esc – 27, клавиша пробела – 32, точка – 46, запятая – 44, вопросительный знак – 63, восклицательный знак – 33 и т. д.

Порядок выполнения работы

1. Уточните номер своего варианта.
2. Составьте блок-схему алгоритмов заданий варианта.
3. Напишите программу по составленной блок-схеме.
4. Отладьте программу (исключите все сообщения об ошибках) и подберите значения для тестирования программ (запишите результаты тестирования).
5. В текстовом редакторе WORD или рукописно оформите отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.
2. Задания варианта

3. Программные единицы.

4. Результат выполнения программы.

Варианты задач

	Вариант 1
1.	Напишите программу, подсчитывающую количество букв во введенном с клавиатуры слове. Ввод осуществляйте в цикле **while do**. Выход из программы – строка '999'.
2.	Дан текст, в котором слова разделены одним или несколькими пробелами. Составить программу, определяющую количество слов в тексте.

	Вариант 2
1.	Напишите программу, подсчитывающую количество вхождений заданной Вами буквы в веденной строке.
2.	Составьте программу, вычеркивающую каждую гласную русскую букву слова X. Дополнительную строку использовать запрещается.
	Вариант 3
1.	Напишите программу, которая вводит строку и выводит ее, сокращая каждый раз на 1 символ до тех пор, пока в строке не останется 1 символ.
2.	Дан текст, в котором слова разделены пробелами. Найдите самое длинное слово текста.
	Вариант 4
1.	Напишите программу, определяющую число слов в строке. Одно слово от другого отделяется одним пробелом.
2.	В заданном предложении найти пару слов, из которых одно является обращением другого.
	Вариант 5
1.	Составьте программу, определяющую, является ли введенное слово числом.
2.	Вычеркните из слова X те буквы, которые встречаются в слове Z дважды. Дополнительную строку использовать запрещается.
	Вариант 6
1.	Введите 2 целых числа. Преобразуйте числа в две строки, объедините их в одну строку и выведите на экран результат.
2.	Вычеркните из заданного слова все буквы, совпадающие с его первой буквой. Дополнительную строку использовать запрещается.
	Вариант 7
1.	Напишите программу, которая удаляет из строки любой введенный с клавиатуры символ.
2.	Дан текст, в котором слова разделены пробелами. Найдите самое короткое слово текста.
	Вариант 8
1.	Составьте программу, удаляющую все пробелы из введенной строки. Примените в ней оператор **Repeat** и функцию **Pos.**
2.	Напишите программу, сортирующую символы введенной с клавиатуры строки в порядке возрастания их номеров в ASCII-таблице. Например, если введено: 'CAB', в результате надо получить 'ABC'.
	Вариант 9
1.	В тексте, состоящем из русских букв и заканчивающемся точкой, подсчитайте количество согласных букв.

2.	Составьте программу, удаляющую все буквы «а» из введенной строки. Примените в ней оператор **Repeat** и функцию **Pos**. Дополнительную строку использовать запрещается.
	Вариант 10
1.	Замените в заданном слове все пробелы символом «;».
2.	Вычислите длину самого короткого слова в предложении из трех слов, разделенных пробелами.
	Вариант 11
1.	Сколько букв «у» в слове стоит на нечетных местах?
2.	Выясните, какая из букв встречается в заданном слове чаще.

	Вариант 12
1.	Найти самое длинное симметричное слово заданного предложения.
2.	Составить программу, определяющую, является ли введенное с клавиатуры слово перевертышем. Перевертышем называется слово, которое одинаково читается как с начала, так и с конца, например: ‘шалаш’, ‘казак’.
	Вариант 13
1.	Сколько букв «у» в слове стоит на четных местах?
2.	Составить программу, которая обращает введенное слово, то есть переставляет символы в обратном порядке, например: Петя – ятеП, мама – амам, программа – аммаргоп. Дополнительную строку использовать запрещается.
	Вариант 14
1.	Замените в заданном слове все буквы «о» пробелами.
2.	Удалить из введенной строки подряд идущие одинаковые символы. Дополнительную строку использовать запрещается.
	Вариант 15
1.	В тексте, состоящем из русских букв и заканчивающемся точкой, подсчитайте количество гласных букв.
2.	Составить программу, удаляющую в строке все, что заключено между фигурными скобками и их самих. Дополнительную строку использовать запрещается.
	Вариант 16
1.	Даны два слова. Поменяйте местами буквы этих слов, занимающие одинаковые позиции.
2.	Определить слово, встречающееся в заданном предложении несколько раз.
	Вариант 17
1.	Вычеркните из заданного слова все буквы, совпадающие с его первой буквой.
2.	Заданы фамилия, имя и отчество учащегося, разделенные пробелом. Напечатайте фамилию и инициалы. Дополнительную строку использовать запрещается.
	Вариант 18
1.	В заданном предложении найти количество таких слов, которые начинаются на букву «а». Слова разделены одним или несколькими пробелами.
2.	Вычеркните *i*-ю букву слова. Дополнительную строку использовать запрещается.
	Вариант 19
1.	Составьте программу перевода строки строчных русских букв в прописные.
2.	В заданном предложении найти слово, содержащее наибольшее количество

	русских гласных букв.
	Вариант 20
1.	В заданном предложении найти количество таких слов, которые кончаются на букву «а».
2.	Задан текст, состоящий из слов, которые разделены одним или несколькими пробелами. Сформируйте новый текст, включив в него слова заданного текста, разделенные только одним пробелом. Дополнительную строку использовать запрещается.
	Вариант 21
1.	Сложное слово состоит из двух частей одинаковой длины и соединительной гласной. Найдите обе части этого слова.
2.	Определить, какие буквы встречаются в слове не менее двух раз.
	Вариант 22
1.	Вычеркните из заданного слова все буквы, совпадавшие с его последней буквой. Дополнительную строку использовать запрещается.
2.	Проверить, имеется ли в заданном тексте баланс открывающих и закрывающих скобок, то есть верно ли, что открывающая скобка всегда предшествует соответствующей закрывающей.
	Вариант 23
1.	Вычеркните из слова X те буквы, которые встречаются в слове Z.
2.	Для заданного текста определить длину содержащейся в нем максимальной серии символов, отличных от букв.
	Вариант 24
1.	Подсчитайте число различных букв в слове.
2.	Составить программу преобразования введенного числа в строку, в которой недостающие разряды числа представлены символом «0» (максимальное количество разрядов 6), например: 4235 – 004235. Дополнительную строку использовать запрещается.
	Вариант 25
1.	Составьте программу подсчета количества повторений заданного фрагмента (цепочки символов). Например, в тексте «банан упал на барабан» фрагмент «ба» встречается 2 раза.
2.	Составьте программу, которая по названию числа <1000, написанному на русском языке, формирует его цифровую запись.
	Вариант 26
1.	В каждом слове введенного предложения поменять местами первую и последнюю буквы.
2.	Перечислить все слова заданного предложения, которые состоят из тех же букв, что и первое слово предложения.
	Вариант 27
1.	Составьте программу шифрования текстового сообщения. Можно использовать такой способ шифрования. Шифровальщик задает ключ шифровки – целое число, – который определяет величину смещения букв русского алфавита, например: ключ = 3, тогда в тексте буква «а» заменяется на «г» и т. д. Используются все буквы русского алфавита. Е считается дважды.
2.	Составьте программу дешифровки текстового сообщения.
	Вариант 28
1.	Составьте программу, выясняющую, на гласную или согласную букву начинается слово X.
2.	Даны два слова. Составьте программу, определяющую, можно или нет из букв слова A составить слово B.

	Вариант 29
1.	Составьте программу перевода строки прописных русских букв в строчные.
2.	Составьте программу подсчета числа одинаковых букв, стоящих на одних и тех же местах в словах X и Y.
	Вариант 30
1.	Составьте программу, вычеркивающую каждую третью букву слова X. Дополнительную строку использовать запрещается.
2.	Составьте программу вычисления суммы мест, на которых в слове X стоят буквы «В» и «П».

Решение типового варианта

Рассмотрим пример оформления работы.

Пусть вариант студента 31.

	Вариант 31
1.	Составьте программу перевода первой строчной буквы каждого слова в прописную. Слова в тексте отделены друг от друга одним пробелом.
2.	Составьте программу преобразования введенного числа в строку, в которой группы из трех символов отделены друг от друга пробелом, например: 12345678 ⇒ 12 345 678. Дополнительную строку использовать запрещается.

Сформируем отчет к лабораторной работе 8.

1. Титульный лист.

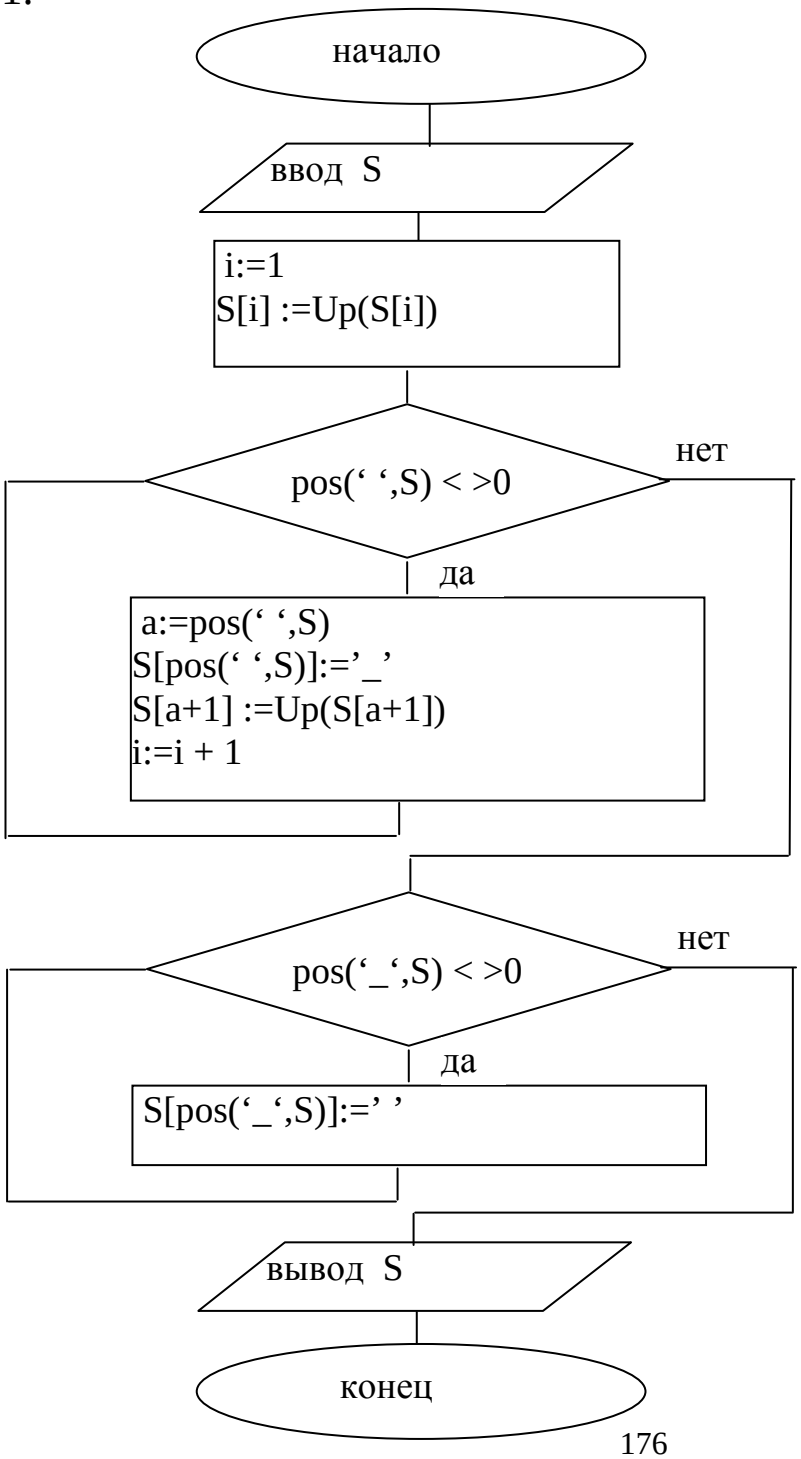
2. Сформулируем задание варианта 31.

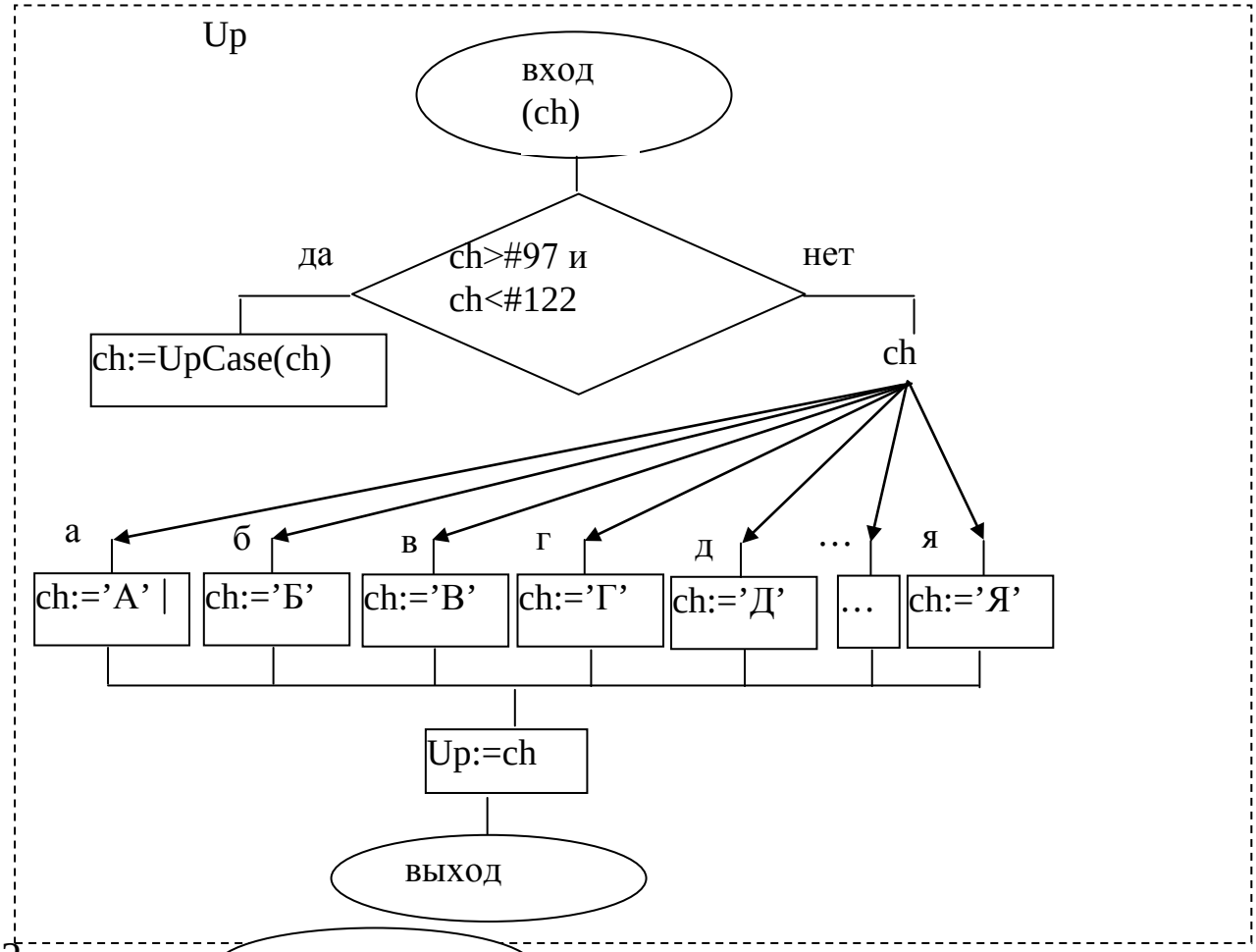
1. Составьте программу перевода первой строчной буквы каждого слова в прописную. Слова в тексте отделены друг от друга одним пробелом.

2. Составьте программу преобразования введенного числа в строку, в которой группы из трех символов отделены друг от друга пробелом, например: 12345678 → 12 345 678. Дополнительную строку использовать запрещается.

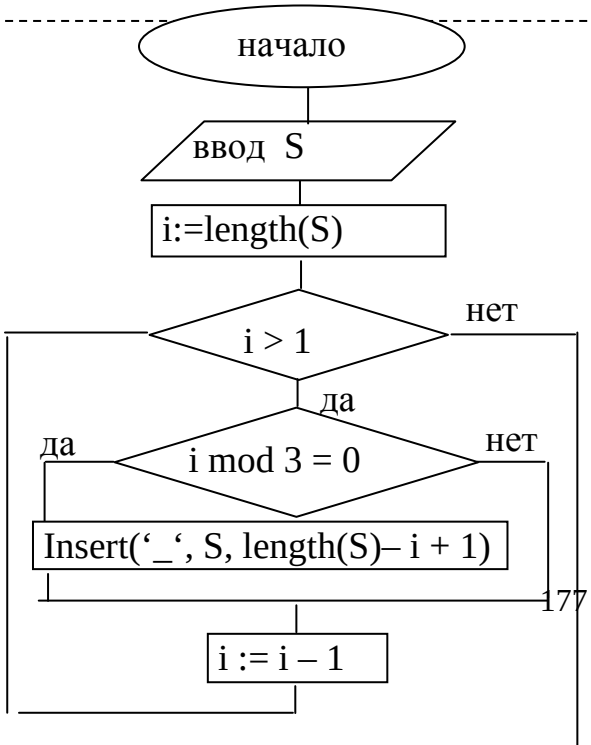
3. Составим блок-схему алгоритмов заданий варианта.

1.




Up
вход
(ch)
да
ch>#97 и
ch<#122
нет
ch:=UpCase(ch)
ch
а
б
в
г
д
…
я
ch:=’А’
ch:=’Б’
ch:=’В’
ch:=’Г’
ch:=’Д’
…
ch:=’Я’
Up:=ch
выход

2.


начало
ввод S
i:=length(S)
i > 1
нет
да
да
i mod 3 = 0
нет
Insert(‘_‘, S, length(S)– i + 1)
i := i – 1

4. Напишем программу по составленным блок-схемам.

```
(*1*)
uses crt;
var
  s:string;
  a,i:byte;
function Up(ch:char):char;
begin
if (ch>#97) and (ch<#122) then ch:=UpCase(Ch)
else case ch of
  'а':ch:='A';'б':ch:='Б';'в':ch:='В';'г':ch:='Г';
  'д':ch:='Д';'е':ch:='Е';'е':ch:='Е';'ж':ch:='Ж';
  'з':ch:='З';'и':ch:='И';'й':ch:='Й';'к':ch:='К';
  'л':ch:='Л'; м':ch:='М';'н':ch:='Н';'о':ch:='О';
  'п':ch:='П';'р':ch:='Р';'с':ch:='С';'т':ch:='Т';
  'ф':ch:='Ф';'х':ch:='Х';'ц':ch:='Ц';'ч':ch:='Ч';
  'ш':ch:='Ш';'щ':ch:='Щ';'ы':ch:='Ы';'э':ch:='Э';
  'ю':ch:='Ю';'я':ch:='Я';
end;
Up:=ch;
end;
begin
  clrscr;
  writeln('Wwod S=');
  readln(s);
{  s:='nauka umeet mnogo gitik';}
```

```
{  s:='наука умеет много гитик';}
  writeln(S);
  i:=1;
  s[i]:=Up(s[i]);
  while pos(' ',s)<>0 do begin
    a:=pos(' ',s);
    s[pos(' ',s)]:='_';
    s[a+1]:=Up(s[a+1]);
    i:=i+1
  end;
  writeln(S);
  while pos('_',s)<>0 do
    s[pos('_',s)]:=' ';
  writeln(S);
  readkey
end.
(*2*)
uses crt;
var
  s:string;
  a,i:byte;
begin
  clrscr;
  writeln('Wwod S=');
  readln(s);
  writeln(S);
  i:=length(s);
  while i>1 do begin
    if i mod 3=0 then insert(' ',s,length(s)-i+1);
    i:=i-1
  end;
  writeln(S);
  readkey
```

```
end.
```

5. Представленные программы не содержат ошибок, подберем значения для тестирования программ.

1. а) S = ‘nauka umeet mnogo gitik’
   S = ‘Nauka Umeet Mnogo Gitik’
   б) S = ‘наука умеет много гитик’
   S = ‘Наука Умеет Много Гитик’
2. а) S = 12345678 S = 12 345 678
   б) S = 123456 S = 123 456

6. В текстовом редакторе WORD или рукописно оформить отчет в соответствии с содержанием.

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1 .Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Юркин, А. Г. Задачник по программированию / А. Г. Юркин. - СПб. : Питер, 2002. – 192 с.

3. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

4. Епанешников, А. Программирование в среде Turbo Pascal 7.0 / А. Епанешников, В. Епанешников. – М. : «ДИАЛОГ-МИФИ», 1993. – 288 с.

5. Немюгин, С. А. Turbo Pascal : практикум / С. А. Немюгин. – СПб. : Питер, 2001 – 256 с.

6. Немюгин, С. А. Turbo Pascal : учебник / С. А. Немюгин. – СПб. : Питер, 2001 – 496 с.

7. Культин, Н. Б. Программирование в Turbo Pascal 7.0 и Delphi / Н. Б. Культин. – 2-е изд., перераб. и доп. – СПб. : БХВ–Петербург, 2001. – 416 с.

6. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

Вопросы для самопроверки

1. Охарактеризуйте тип данных CHAR.
2. Охарактеризуйте тип данных STRING.
3. Сформулируйте сходство и различие между строкой и одномерным массивом.
4. Формат процедуры работы со строками DELETE выглядит следующим образом: **PROCEDURE DELETE (VAR S: STRING; INDEX: INTEGER; COUNT: INTEGER);** Расшифруйте каждое слово этого формата.
5. Формат процедуры работы со строками INSERT выглядит следующим образом: **PROCEDURE INSERT(SOURCE: STRING; VAR S: STRING; INDEX: INTEGER);** Расшифруйте каждое слово этого формата.
6. Формат процедуры работы со строками STR выглядит следующим образом: **PROCEDURE STR(X [: M [: N]]; VAR S: <СТРОКОВЫЙ ТИП>);** Расшифруйте каждое слово этого формата.
7. Формат процедуры работы со строками VAL выглядит следующим образом**: PROCEDURE VAL(S: <СТРОКОВЫЙ ТИП>; VAR V; VAR CODE: INTEGER);** Расшифруйте каждое слово этого формата.
8. Формат функции работы со строками CONCAT выглядит следующим образом: **FUNCTION CONCAT(SL [, S2, ..., SN]: STRING): STRING;** Расшифруйте каждое слово этого формата.
9. Формат функции работы со строками COPY выглядит следующим образом **FUNCTION COPY(S: STRING; INDEX: INTEGER; COUNT: INTEGER): STRING;** Расшифруйте каждое слово этого формата.
10. Формат функции работы со строками LENGTH выглядит следующим образом **FUNCTION LENGTH(S: STRING): INTEGER;** Расшифруйте каждое слово этого формата.

11. Формат функции работы со строками POS выглядит следующим образом **FUNCTION POS(SUBSTR,S: STRING): BYTE;** Расшифруйте каждое слово этого формата.

12. Формат функции преобразования типов CHR выглядит следующим образом **FUNCTION CHR(X: BYTE): CHAR;** Расшифруйте каждое слово этого формата.

13. Формат функции преобразования типов ORD выглядит следующим образом **FUNCTION ORD(X: <ПОРЯДКОВЫЙ ТИП>): LONGINT;** Расшифруйте каждое слово этого формата.

14. Дана строка S. Замените в ней каждую из групп стоящих рядом точек одной точкой.

15. Дана строка S. Заменить каждую точку многоточием, то есть тремя точками.

16. Дана строка S =’информатика’. Используя цикл и оператор присваивания сформировать STR = ‘форма’.

17. Дана строка S = ’индивидуальность’. Используя четыре оператора присваивания, сформировать STR = ‘диво’.

ЛАБОРАТОРНАЯ РАБОТА 9. РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ВВОДА – ВЫВОДА В ТУРБО ПАСКАЛЕ. РАБОТА С ФАЙЛАМИ (2 ЧАСА)

Цель работы: выявить особенности взаимодействия системы Турбо Паскаль с внешними носителями. Научиться создавать и использовать файлы последовательного и прямого доступа в системе Турбо Паскаль.

Теоретические положения

Работа с файлами

Удобным способом сохранения информации, полученной в ходе выполнения программы, служит запись этой информации на внешний носитель. Сейчас довольно трудно сформулировать полный список разновидностей таких носителей (например: твердые диски, гибкие диски (дискеты), магнитные ленты, и так далее, и тому подобное).

Такой список бесполезен, потому что сама система программирования умеет учитывать нюансы внешнего носителя. К тому же, во-первых, этот список постоянно расширяется, во-вторых, обработка файла стандартизирована, то есть для новых носителей не требуется новых понятий, процедур и функций. Запись особенно желательна, если объем информации велик и если в дальнейшем предполагается использовать эту информацию в других программах.

В Паскале предусмотрены специальные объекты – ФАЙЛЫ. Объясняя принципы работы с файлами, следует для простоты и наглядности считать, что каждый файл записан на некой собственной ленте и при этом с самого начала ленты записано имя файла (идентификатор). После имени файла записаны компоненты, а вслед за самой последней компонентой записан признак конца файла. Еще одно важное виртуальное понятие – указатель текущей позиции в файле. Он определяет текущую позицию в файле, то есть компоненту, которую можно обработать следующей операцией. У разнотипных файлов главная общая особенность – любое считывание или запись автоматически переводит указатель в следующую позицию.

Под файлом понимается либо именованная область внешней памяти персонального компьютера (жесткого диска, гибкой дискеты, электронного «виртуального» диска), либо логическое устройство – потенциальный источник или приемник информации.

Любой файл имеет три характерные особенности.

1. У файла есть имя, что дает возможность программе работать одновременно с несколькими файлами.

2. Файл содержит компоненты одного типа. Типом компонентов может быть любой тип, кроме файлов. Иными словами, нельзя создать «файл файлов».

3. Длина файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти.

Файловый тип или переменную файлового типа можно задать одним из трех способов:

• типизированные файлы задаются предложением <имя> = **FILE OF** <тип>;

• текстовые файлы определяются предложением <имя> = TEXT;

• нетипизированные файлы определяются предложением <имя> = **FILE**;

Пример.

```
type
  text80 = file of string[80];
var
  f1: file of char;
  f2: text;
  f3: file;
  f4: text80;
  f5: file of string[10];
```

В этом примере f1, f4, f5 – типизированные файлы, f2 – текстовый файл, f3 – нетипизированный файл.

Вид файла, вообще говоря, определяет способ хранения информации в файле. Однако в Турбо Паскале нет средств контроля вида ранее созданных файлов. При объявлении уже существующих файлов программист должен сам следить за соответствием вида объявления характеру файла.

Доступ к файлам

Любой программе доступны два предварительно объявленных файла со стандартными файловыми переменными: INPUT – для чтения данных с клавиатуры и OUTPUT – для вывода на экран. Стандартный Паскаль требует обязательного упоминания этих файлов в заголовке. Турбо Паскаль открывает эти устройства автоматически, для него это заранее объявленные переменные типа TEXT.

Два следующих примера работают совершенно одинаково

Пример 1. Дан целочисленный поток данных, завершающийся значением 0. Просуммировать компоненты, кратные семи.

```
uses crt;
var
  a,s: integer;
begin
  clrscr;
  reset(input);    {Объявлено чтение данных}
  repeat
    readln(input,a);
    if a mod 7 = 0 then s := s + a
  until (a = 0);
  writeln(output, s);
  readkey
end.
```

Пример 2. Дан целочисленный поток данных, завершающийся значением 0. Просуммировать компоненты, кратные семи.

```
uses crt;
var
  a,s: integer;
begin
  clrscr;
  repeat
    readln(a);
    if a mod 7 = 0 then s := s + a
  until (a = 0);
  writeln(s);
  readkey
end.
```

INPUT и OUTPUT не служебные слова, а по умолчанию определенные идентификаторы, то есть их можно использовать как обыкновенные файловые переменные. Именно так сделано в примере 3, который использует еще не прописанные нами процедуры ASSIGN, RESET, CLOSE.

Пример 3. Дан целочисленный поток данных, завершающийся значением 0. Просуммировать компоненты, кратные семи.

```
uses crt;
var
  a,s: integer;
begin
  clrscr;
  assign(input, 'primer_9.pas'); {файловая переменная связывается
                                  с файлом 'primer_9.pas'}
  reset(input); {объявлено чтение данных из файла}
  repeat
    readln(input, a); {чтение значения из файла в переменную a}
    if a mod 7 = 0 then s := s + a
  until (a = 0);
  writeln('S=', s);
  close(input); {закрывает файл, то есть очищает буфер, а значит,
                 действительно записывает новые данные на
                 диск}
  readkey
end.
```

Любые файлы, а также логические устройства становятся доступны программе только после выполнения особой процедуры открытия файла (логического устройства). Эта процедура заключается в связывании ранее объявленной переменной с именем существующего или вновь создаваемого файла, а также в указании направления обмена информацией: чтение из файла или запись в него.

Общие процедуры и функции для всех видов файлов
Текстовые файлы

Процедура **ASSIGN.** Файловая переменная связывается с именем файла**.**

Формат обращения: ASSIGN (<ф.п.>,<имя файла или л.у.>)

Если имя файла задается в виде пустой строки, например ASSIGN (f, ’’), то в зависимости от направления обмена данными файловая переменная связывается со стандартным оператором Input или Output.

Процедура **CLOSE**. Закрывает файл, однако связь файлов переменной с именем файла, установленная ранее процедурой ASSIGN, сохраняется.

Формат обращения: CLOSE (< ф.п.>)

Процедура **ERASE.** Уничтожение внешнего файла. Удаляется внешний файл, с которым связана файловая переменная.

Формат обращения: ERASE (<ф.п.>)

Процедура **RENAME.** Переименовывает файл.

Формат обращения: RENAME (<ф.п.>, <новое имя>)

Здесь <новое имя> – строковое выражение, содержащее новое имя файла.

Процедура **RESET.** Открытие существующего файла.

Открывается существующий файл, с которым связана файловая переменная, и указатель текущей компоненты файла настраивается на начало файла.

Формат обращения: RESET (<ф.п.>)

Процедура **REWRITE.** Открывает новый файл.

Открывается новый пустой файл, и ему присваивается имя, заданное процедрой Assign. Если файл с таким именем уже существует, то он очищается.

Формат обращения: REWRITE (<ф.п.>)

Функция **EOF.** Логическая функция, тестирующая конец файла. Возвращает TRUE, если файловый указатель стоит в конце файла. При записи это означает, что очередной компонент будет добавлен в конце файла, при чтении – что файл исчерпан.

Формат обращения: EOF (<ф.п.>).

Процедура **APPEND.** Открывает файл для добавления в конец информации. Открывается существующий файл, с которым связана

файловая переменная, и указатель текущей компоненты файла настраивается на конец файла.
Формат обращения: APPEND (<ф.п.>).

Процедура **READ.** Обеспечивает ввод символов, строк и чисел.
Формат обращения: READ (<Ф.п.>, < сп. ввода>)
или READ (<сп. ввода>)

Процедура **READLN.** Обеспечивает ввод символов, строк и чисел. Эта процедура полностью идентична процедуре READ за исключением того, что выводимая строка символов завершается кодами CR и LF.

Процедура **WRITE.** Обеспечивает вывод информации в текстовый файл или передачу ее на логическое устройство.
Формат обращения: WRITE(<файл>, <сп.вывода>)
или WRITE(<сп.вывода>)

Процедура **WRITELN.** Обеспечивает вывод информации в текстовый файл или передачу ее на логическое устройство. Эта процедура полностью идентична процедуре WRITE за исключением того, что выводимая строка символов завершается кодами CR и LF.

Логическая функция EOLN. Логическая функция, тестирующая конец строки. Возвращает TRUE, если во входном текстовом файле достигнут маркер конца строки.
Формат обращения: EOLN (<ф.п.>)

В формулировке задач на работу с файлами чаще всего установка «Дан файл». Для корректного представления алгоритма решения задачи нужно, чтобы был файл. Учащийся должен решить технический момент его создания.

Когда речь идет о текстовом файле, можно использовать способы создания файла:

1) вставить в алгоритм решения задачи алгоритм создания файла.

а) ввод вручную данных с клавиатуры,

б) автоматический ввод через счетчик случайных значений,

2) ввод вручную с клавиатуры в текстовый редактор.

Рассмотрим подробнее пример каждого способа.

1а) Пример. Создать целочисленный файл f из чисел в диапазоне больше -10, и меньше 20, в количестве 10 штук. Данные введите вручную с клавиатуры.

Решение: Текстовый файл будет создан под именем 'proba.pas' в каталоге, являющемся активным. Если есть желание, что бы файл располагался в конкретной папке, то его надо указать явно, например: assign(f, 'c:\students\kurs1\fedja\proba.pas');

```
uses crt;
var
  f: text;
  a,i: integer;
begin
  assign(f, 'proba.pas');
  rewrite(f);
  i:=1;
  repeat
    repeat
      write('Введи ',i,'-ое число:');
      readln(a);
  until (a >= –10) and (a <= 20);
  i := i + 1;
  writeln(f, a)
 until i > 10;
 reset(f);
 while not eof(f) do begin
  readln(f, a);
  writeln(a)
 end;
 close(f)
end.
```

1б) Пример. Создать целочисленный файл f из чисел в диапазоне больше – 10, и меньше 20, в количестве 10 штук. Данные введите автоматически через счетчик случайных чисел.

```
uses crt;
var
  f: text;
  a, i: integer;
begin
  randomize;
  clrscr;
  assign(f, 'proba.pas');
  rewrite(f);
  i := 1;
  for i := 1 to 10 do begin
    a := random(30) – 10;
    writeln(i, '-oe число=', a);
    writeln(f, a)
  end;
  close(f);
 readkey
end.
```

2) Текстовый редактор следует использовать любой поддерживающий ASCII (DOS) кодировку (Word, WordPad, стандартный блокнот WINDOWS). Этот способ в работе выглядит следующим образом:

I. Создали файл в текстовом редакторе, сохранили его на диске, запомнили имя и расположение, вышли из редактора.

II. Запустили систему Турбо Паскаль, составили программу.

Недостатком такого способа является необходимость, во-первых, знания кодировки ASCII для используемого редактора (она нередко бывает неоднозначна в обозначении), во-вторых, как прави-

ло, многократного возврата в текстовый редактор для уточнения расположения и внешнего вида данных.

Можно рекомендовать воспользоваться редактором самого Турбо Паскаля: то есть открыть опцией NEW новое окно, ввести данные. Сохранить файл с расширением PAS. Для хранения данных на диске это безразлично, а при загрузке следующего сеанса Турбо Паскаль без дополнительного ввода маски файлов выведет как имена файлов-программ, так и файлов данных.

Текстовый файл способен хранить как символьные, так и числовые данные. Часто создаваемый файл должен хранить данные в особом, специально оговоренном заданием формате, что демонстрирует следующий пример.

Пример.

Создать текстовый файл из фамилий, имен и дат рождения. Данные вводить с клавиатуры, дату рождения представить в виде дд/мм/гггг

```
uses
  crt;
var
  f:text;
  st, st1:string;
  i:integer;
function str_i(i:integer):string;
var
  st:string;
begin
  str(i,st);
  if i < 10 then st :='0' + st;
  str_i := st
end;
procedure input_str(s: string; var st: string);
begin
```

```
  write(s); readln(st);
end;
function Y_N (s: string): boolean;
var
  ch: char;
begin
  writeln(s);
  repeat
     ch := readkey
  until (ch = 'Y') or (ch = 'y') or (ch = 'N') or (ch = 'n');
  if (ch = 'Y') or (ch = 'y') then Y_N := true
                              else Y_N := false
end;
begin
  assign(f, 'proba.pas');
  rewrite(f);
  while Y_N('Вводим данные ? ') do begin
    input_str('Введи фамилию ',st);
    write(f,st+' ');
    input_str('Введи имя ',st);
    write(f,st+' ');
    writeln('Введи день рождения');
    readln(i);
    st := str_i(i) + '/';
    writeln('Введи месяц рождения');
    readln(i);
    st := st + str_i(i) + '/';
   writeln('Введи год рождения');
   readln(i);
    st := st + str_i(i);
    writeln(f, st);
  end;
```

```
  close(f)
end.
```

Типизированные файлы

Длина любого компонента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому из них (то есть доступ к компоненту по его порядковому номеру).

Перед первым обращением к процедурам ввода/вывода указатель стоит в его начале и указывает на первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные в списках ввода/вывода должны иметь тот же тип, что и компоненты файла. Если этих переменных в списке несколько, указатель будет смещаться после каждой операции обмена данными между переменными и дисковым файлом.

Процедура **ASSIGN.** Файловая переменная связывается с именем файла**.**

Формат обращения: ASSIGN (<ф.п.>,<имя файла или л.у.>)

Если имя файла задается в виде пустой строки, например ASSIGN (f, ’’), то в зависимости от направления обмена данными файловая переменная связывается со стандартным оператором Input или Output.

Процедура **CLOSE**. Закрывает файл, однако связь файлов переменной с именем файла, установленная ранее процедурой ASSIGN, сохраняется.

Формат обращения: CLOSE (< ф.п.>)

Процедура **ERASE.** Уничтожение внешнего файла. Удаляется внешний файл, с которым связана файловая переменная.

Формат обращения: ERASE (<ф.п.>)

Процедура **RENAME.** Переименовывает файл.

Формат обращения: RENAME (<ф.п.>, <новое имя>)

Здесь <новое имя> – строковое выражение, содержащее новое имя файла.

Процедура **TRUNCATE.** Удаление компонентов из файла начиная с текущей позиции и до конца файла.

Формат обращения: TRUNCATE (<ф.п.>)

Процедура **RESET.** Открытие существующего файла.

Открывается существующий файл, с которым связана файловая переменная, и указатель текущей компоненты файла настраивается на начало файла.

Формат обращения: RESET (<ф.п.>)

Процедура **REWRITE.** Открывает новый файл.

Открывается новый пустой файл, и ему присваивается имя, заданное процедурой Assign. Если файл с таким именем уже существует, то он очищается.

Формат обращения: REWRITE (<ф.п.>)

Функция **EOF.** Логическая функция, тестирующая конец файла. Возвращает TRUE, если файловый указатель стоит в конце файла. При записи это означает, что очередной компонент будет добавлен в конце файла, при чтении – что файл исчерпан.

Формат обращения: EOF (<ф.п.>).

Процедура **READ.** Обеспечивает чтение очередных компонентов типизированного файла.

Формат обращения: READ (<Ф.п.>,< сп. ввода>)

Файловая переменная должна быть объявлена предложением **file of...** и связана с именем файла процедурой **assign.** Файл необходимо открыть процедурой **reset**. Если файл исчерпан, обращение к **READ** вызовет ошибку ввода/вывода.

Процедура **WRITE.** Используется для записи данных в типизированный файл.

Формат обращения: WRITE(<ф.п.>, <сп.вывода>)

Процедура **SEEK.** Смещает указатель к требуемому компоненту. Формат обращения: SEEK(<ф.п.>,<номер компонента>)

Функция **EILESIZE.** Возвращает значение типа **longint**, которое содержит количество компонентов файла.

Формат обращения: EILESIZE(<ф.п.>)

Пример.

Переставить указатель в конец файла.

seek(f,FileSize(f));

где f – файловая переменная.

Функция **EILEPOS.** Возвращает значение типа **longint**, содержащее порядковый номер компонента файла, который будет обрабатываться следующей операцией ввода/вывода.

Формат обращения: EILEPOS(<ф.п.>)

Типизированный файл, созданный программно, может быть загружен в любой текстовый редактор, но его содержимое в кодах ASCII плохо распознаваемо и фактически нередактируемо. Исключением является файл типа CHAR, поэтому рекомендуется задания движения компонент в файле выполнять на этом типе, тогда результат очевиден при имеющемся окне с содержимым файла. В любых остальных случаях следует обязательно выводить на экран содержимое файла до и после сделанных преобразований.

Рассмотрим примеры самых частых ситуаций программной обработки файлов.

Пример 1. Создать файл, компонентами которого являются 20 действительных чисел в диапазоне от 1 до 5000.

```
var
  f: file of real;
  i: byte;
  r: real;
begin
  assign(f, 'proba.pas');
  rewrite(f);
  randomize;
  for i:=1 to 20 do begin
    r := random * 5000 + 1;
```

```
    write(f, r)
  end;
  close(f);
end.
```

Пример 2. Компонентами файла f являются действительные числа. Найти сумму компонент файла.

```
uses crt;
var
  f: file of real;
  re,s: real;
begin
  assign(f, 'proba.pas');
  reset(f);
  while not eof(f) do begin
    read(f, re);
    s := s + re
  end;
  close(f);
  writeln(s: 10: 3);
  readkey
End.
```

Пример 3. Компонентами файла f являются действительные числа. Найти последнюю компоненту файла.

```
uses crt;
var
  f: file of real;
  re: real;
begin
  assign(f, 'proba.pas');
  reset(f);
  seek(f, filesize(f) – 1);
  read(f, re);
```

```
  writeln(re);
  close(f);
  readkey
End.
```

Пример 4. Инвертировать, то есть расставить компоненты в обратном порядке, типизированный файл, без использования промежуточного файла.

```
uses crt;
var
  f: file of real;
  re1,re2: real;
  i, n: word;
procedure wywod;
var
  re: real;
begin
  reset(f);
  while not eof(f) do begin
    read(f, re);
    write(re: 10: 2)
  end;
  close(f);
  writeln
end;
begin
  assign(f, 'proba.pas');
  wywod;{Выводим на экран первоначальное
                расположение компонент в файле}
  reset(f);
  n := filesize(f) – 1;
  for i: =0 to n div 2 do begin
    seek(f, i);
```

```
    read(f, re1);
    seek(f, n – i);
    read(f, re2);
    seek(f, i);
    write(f, re2);
    seek(f, n – i);
    write(f, re1);
  end;
  close(f);
  wywod;{Выводим на экран полученное
              расположение компонент в файле}
  readkey
End.
```

Пример 5. Дан символьный типизированный файл f. Написать программу удаляющую из файла все знаки '$'.

```
uses crt;
var
  f, f_tmp: file of char;
  ch:char;
begin
  assign(f, 'proba.pas');
  assign(f_tmp, 'proba1.pas');
  rewrite(f);
  writeln('Введите последовательность символов в файл,
           последний символ ENTER');
  repeat
    ch := readkey;
    write(f, ch);
    write(ch)
  until ch = #13;
  writeln;
  seek(f, 0);
```

```
  rewrite(f_tmp);
  while not eof(f) do begin
    read(f,ch);
    if ch <> '$' then write(f_tmp, ch);
  end;
  erase(f);
  rename(f_tmp,'proba.pas');
  seek(f_tmp,0);
  while not eof(f_tmp) do begin
    read(f_tmp,ch);
    write(ch)
  end;
  readkey
end.
```

Пример 6. Дан символьный типизированный файл f. Написать программу, удаляющую из файла все символы, начиная со знака '$'.

```
uses crt;
var
  f: file of char;
  ch: char;
  i: byte;
  files: string;
begin
  files := 'Деньги – это не только $ но и р.';
  assign(f, 'proba.pas');
  rewrite(f);
  for i:=1 to length(files) do begin
    write(f,files[i]);
  end;
  seek(f,0);
  while not eof(f) do begin
    read(f,ch);
```

```
    if ch = '$' then begin
      seek(f, filepos(f)-1);
      truncate(f)
    end;
  end;
  seek(f,0);
  while not eof(f) do begin
    read(f,ch);
    write(ch)
  end;
  readkey
end.
```

Нетипизированные файлы

Нетипизированные файлы объявляются как файловые переменные типа **file** и отличаются тем, что для них не указан тип компонентов. Отсутствие типа делает эти файлы, с одной стороны, совместимыми с любыми другими файлами, а с другой – позволяет организовать высокоскоростной обмен данными между диском и памятью.

При инициализации нетипизированного файла процедурами **reset** или **rewrite** можно указать длину записи нетипизированного файла в байтах. Например, так:

```
var
 f:file;
.....
assign(f,'proba.dat');
reset(f,512);
```

Длина записи нетипизированного файла указывается вторым параметром при обращении к процедурам **reset** или **rewrite**, в качестве которого может использоваться выражение типа **word**. Если длина записи не указана, она принимается равной 128 байтам.

На длину записи нетипизированного файла накладываются требования положительности и максимальной длины не более 65535

байт. Однако для максимальной скорости обмена данными следует задавать длину, которая была бы кратна длине физического сектора дискового носителя информации (512 байт).

При работе с нетипизированными файлами могут применяться все процедуры и функции, доступные типизированным файлам, за исключением **READ** и **WRITE,** которые заменяются соответственно высокоскоростными процедурами **BLOCKREAD** и **BLOCKWRITE**. Для вызова этих процедур используется следующий формат:

BLOCKREAD(<ф.п.>,<буф.>,<N>,[<NN>]

BLOCKWRITE(<ф.п.>,<буф.>,<N>,[<NN>],

где

<буф.> – буфер, то есть имя переменной, которая будет участвовать в обмене данными с дисками;

<N> – количество записей, которые должны быть прочитаны или записаны за одно обращение к диску;

<NN> – необязательный параметр, содержащий при выходе из процедуры количество фактически обработанных записей.

За одно обращение к процедурам может быть передано до N*RECS байт, где RECS – длина записи нетипизированного файла. Передача идет начиная с первого байта переменной <буф.>. Программист сам заботится о достаточности длины внутреннего представления переменной <буф.> для размещения всех N*RECS байт при чтении информации с диска. Если при чтении указана переменная <буф.> недостаточной длины или если в процессе записи на диске не окажется свободного пространства, возникнет ошибка ввода/вывода, которую можно заблокировать, указав необязательный параметр <NN> (переменная типа **WORD**).

После завершения процедуры указатель смещается на <NN> записей. Процедурами **SEEK, EILESIZE, EILEPOS** можно обеспечить доступ к любой записи нетипизированного файла.

Составление блок-схем для работы с файлом обычно игнорируется в большинстве методических пособий, так как движение указа-

теля в файле блок-схемой никак не отражается. В стандартах оформления алгоритмов приводятся специальные значки, изображающие передачу данных на различные типы внешних носителей, однако чаще всего используют обычное изображение ввода/вывода, только иногда добавляют словесное указание: «в файл F», «из файла F». Мы будем обходиться без блок-схем, так как в работе с файлами главное – контроль движения указателя файла, а его все равно приходится учитывать функциями, которые в блок-схеме не отражаются.

Порядок выполнения работы

1. Уточните номер своего варианта.
2. Напишите программу.
3. Отладьте программу.
4. В текстовом редакторе WORD или рукописно оформить отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.
2. Задания варианта.
3. Программные единицы.

Варианты задач

В заданиях первая задача должна быть сделана с использованием текстового, а вторая – типизированного файла.

	Вариант 1
1.	Дан файл *f*, компоненты которого являются действительными числами. Найти сумму компонент файла *f* и дописать найденное значение в конец файла.
2.	Дан файл *f*, содержащий сведения о книгах. Сведения о каждой из книг – это фамилия автора, название и год издания. Удалить сведения о книгах данного автора, изданных начиная с заданного года.
	Вариант 2.
1.	Дан файл *f*, компоненты которого являются действительными числами. Найти произведение компонент файла *f* и вставить найденное значение в начало файла.
2.	Дан файл *f*, содержащий различные даты. Каждая дата – это число, месяц и год. Удалить из файла год с наименьшим номером.
	Вариант 3

1.	Дан файл *f*, компоненты которого являются действительными числами. Найти сумму квадратов компонент файла *f* и заменить первую компоненту файла найденным значением.
2.	Дан файл *f*, содержащий различные даты. Каждая дата – это число, месяц и год (По образцу: 1/09/08 – 1 сентября 2008 года). Удалить из файла все весенние даты.
	Вариант 4
1.	Дан файл *f*, компоненты которого являются действительными числами. Сравнить модуль суммы и квадрат произведения компонент файла *f*. Большее значение дописать в конец файла.
2.	Дан файл *f*, содержащий сведения о книгах. Сведения о каждой из книг – это фамилия автора, название и год издания. Определить, имеется ли книга с названием «Информатика». Если да, то вывести на экран фамилию автора и год издания.
	Вариант 5
1.	Дан файл *f*, компоненты которого являются действительными числами. Найти последнюю компоненту файла. Если она положительна, заменить ею первую компоненту файла.
2.	Дан файл *f*, содержащий различные даты. Каждая дата – это число, месяц и год (По образцу: 1/09/08 – 1 сентября 2008 года). Удалить из файла самую позднюю дату.
	Вариант 6
1.	Дан файл *f*, компоненты которого являются действительными числами. Дописать файл наибольшим из значений компонент.
2.	Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Дан файл *f*, содержащий сведения о нескольких автомобилях. Найти фамилии владельцев и номера автомобилей данной марки.
	Вариант 7
1.	Дан файл *f*, компоненты которого являются действительными числами. Наименьшее из значений компонент с четными номерами записать на места всех компонент с четными номерами.
2.	Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Дан файл *f*, содержащий сведения о нескольких автомобилях. Найти количество автомобилей каждой марки.
	Вариант 8
1.	Дан файл *f*, компоненты которого являются действительными числами. Найти сумму наибольшего и наименьшего из значений компонент. Дописать найденное значение в конец файла.
2.	Прямая на плоскости задается уравнением $ax + by + c = 0$, где a и b одновременно не равны нулю. Будем рассматривать только прямые, для которых коэффициенты a, b, c – целые числа. Пусть f – файл, содержащий коэффициенты нескольких прямых (не менее трех). Переписать из файла f в файл g коэффициенты всех различных прямых файла f.
	Вариант 9
1.	Дан файл *f*, компоненты которого являются действительными числами. Найти разность первой и последней компонент файла. Заменить полученным значением последнюю компоненту.

2.	Прямая на плоскости задается уравнением $ax + by + c = 0$, где a и b одновременно не равны нулю. Будем рассматривать только прямые, для которых коэффициенты a, b, c – целые числа. Пусть f – файл, содержащий коэффициенты нескольких прямых (не менее трех). Переписать из файла f в файл g коэффициенты тех прямых, которые параллельны первой из прямых, заданной в файле f.
	Вариант 10
1.	Дан файл f, компоненты которого являются целыми числами. Заменить последнюю компоненту количеством четных чисел среди компонент.
2.	Прямая на плоскости задается уравнением $ax + by + c = 0$, где a и b одновременно не равны нулю. Будем рассматривать только прямые, для которых коэффициенты a, b, c – целые числа. Пусть f – файл, содержащий коэффициенты нескольких прямых (не менее трех). Переписать из файла f в файл g коэффициенты тех прямых, которые пересекают первую из прямых, заданных в файле f.
	Вариант 11
1.	Дан файл f, компоненты которого являются целыми числами. Дописать в файл количество удвоенных нечетных чисел среди компонент.
2.	Даны два символьных файла f и g. Файл f содержит произвольный текст. Файл g содержит не более 10 символов, которые разделены запятыми. Эти символы образуют пары: каждый первый символ считается заменяемым, каждый второй символ – заменяющим. Найти в файле f все заменяемые символы и заменить их на соответствующие заменяющие.
	Вариант 12
1.	Дан файл f, компоненты которого являются целыми числами. Дописать в файл количество квадратов нечетных чисел среди компонент.
2.	Дан символьный файл f, содержащий произвольный текст. Слова в тексте разделены пробелами и знаками препинания. Получить 5 наиболее часто встречающихся слов и число их появлений. Решить задачу без ограничения на длину слов.
	Вариант 13
1.	Последовательность $x_1, x_2, \ldots$ образована по закону $x_i = \dfrac{i - 0{,}1}{i^3 + \lvert\operatorname{tg} 2i\rvert}$ $(i = 1,2,\ldots)$. Дано действительное $\varepsilon > 0$. Записать в файл h члены последовательности $x_1, x_2, \ldots$, остановившись после первого члена, для которого выполнено $\lvert x_i\rvert < \varepsilon$.
2.	Дан символьный файл f. Считая, что количество символов в слове не превосходит двадцати, определить, сколько в файле f имеется слов, состоящих из одного, двух, трех и так далее символов. Считать, что слова отделены друг от друга одним пробелом.
	Вариант 14
1.	Последовательность $x_1, x_2, \ldots$ образована по закону $x_i = i!$ (i факториал) ($i = 1, 2, \ldots$ Дано действительное $n > 0$. Записать в файл h члены последовательности $x_1, x_2, \ldots$, остановившись после первого члена, для которого выполнено $x_i > n$.
2.	Дан символьный файл f. Считая, что количество символов в слове не превосходит двадцати, определить количество слов в файле f.

	Вариант 15
1.	Последовательность x_1, x_2, … образована по закону $x_i = 2^i + 3^i$ (i = 1, 2, …). Дано действительное $n > 0$. Записать в файл *h* члены последовательности x_1, x_2, …, остановившись после первого члена, для которого выполнено $x_i > n$.
2.	Дан символьный файл *f*. Найти самое длинное слово среди слов, вторая буква которых *e*; если слов с наибольшей длиной несколько, то найти последнее. Если таких слов нет вообще, то сообщить об этом.
	Вариант 16
1.	Последовательность x_1, x_2, … образована по закону $x_i = \frac{2^i}{i!}$ (i = 1, 2, …). Дано действительное $\varepsilon > 0$. Записать в файл *h* члены последовательности x_1, x_2, …, остановившись после первого члена, для которого выполнено $x_i < \varepsilon$.
2.	Дан символьный файл *f*. Группы символов, разделенные одним пробелом и не содержащие пробелов внутри себя, будем называть словами. Удалить из файла все однобуквенные слова.
	Вариант 17
1.	Последовательность x_1, x_2, … образована по закону $x_i = 1 + \frac{1}{2} + \ldots + \frac{1}{i}$ (i = 1, 2, …). Дано действительное $n > 0$. Записать в файл *h* члены последовательности x_1, x_2, …, остановившись после первого члена, для которого выполнено $x_i > n$.
2.	Дан символьный файл *f*. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Удалить из файла все лишние пробелы, то есть повторяющиеся пробелы и пробелы в начале и конце строк.
	Вариант 18
1.	Последовательность x_1, x_2, … образована по закону $x_i = 1 - \frac{1}{2} + \ldots + \frac{(-1)^{i+1}}{i}$ (i = 1, 2, …). Дано действительное $\varepsilon > 0$. Записать в файл *h* члены последовательности x_1, x_2, … остановившись после первого члена, для которого выполнено $\lvert x_i - x_{i-1} \rvert < \varepsilon$.
2.	Дан символьный файл *f*. Записать в файл *g* с сохранением порядка следования те символы файла *f*, которым в этом файле предшествует буква *a*, и затем те, вслед за которым в этом файле идет буква *a*.
	Вариант 19
1.	Дан символьный файл *f*. Получить файл *g*, образованный из файла *f*, записывая первую компоненту 1 раз, вторую 2 раза, третью 3 раза и так далее.
2.	Даны символьные файлы *f* и *g*. Определить, совпадают ли компоненты файла *f* с компонентами файла *g*. Если нет, то получить номер первой компоненты, в которой файлы *f* и *g* отличаются между собой. В случае, когда один из файлов имеет *n* компонент (n ≥ 0) и повторяет начало другого (более длинного) файла, ответом должно быть число *n* + 1.
	Вариант 20
1.	Даны символьные файлы *f1* и *f2*. Переписать с сохранением порядка следования компоненты файла *f1* в файл *f2*, а компоненты файла *f2* – в файл *f1*.
2.	Дан символьный файл *f*. Подсчитать число вхождений в файл сочетаний *ab*.

	Вариант 21
1.	Даны файлы *f1*, *f2*, *f3*, *f4*, *f5*, компоненты которых являются действительными числами. Организовать обмен компонентами между файлами в соответствии со следующей схемой: *f1* *f2* *f3* *f4* *f5* ↓ ↓ ↓ ↓ ↓ *f3* *f4* *f5* *f1* *f2* то есть компоненты файла *f1* переписываются в файл *f3*, компоненты файла *f2* – в *f4* и так далее.
2.	Дан символьный файл *f*. Подсчитать число вхождений в файл каждой из букв *a*, *b*, *c*, *d*, *e*, *f* и вывести результат в виде таблицы $a - N_a \quad b - N_b \quad c - N_c$ $d - N_d \quad e - N_e \quad f - N_f$ где N_a, N_b, N_c, N_d, N_e, N_f – числа вхождений соответствующих букв.
	Вариант 22
1.	Дан символьный файл *f*. В файле *f* не менее двух компонент. Определить, являются ли два первых символа файла цифрами.
2.	Дан символьный файл *f*. Если последовательность символов *END* встречается в файле хотя бы один раз, то добавить в его конец символы *e*, *n*, *d*.
	Вариант 23
1.	Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* квадраты всех компонент файла *f*, являющиеся четными числами.
2.	Дан файл *f*, количество целочисленных компонент которого кратно 7. Записать в файл *g* наименьшее значение первых семи компонент файла *f*, затем – следующих семи компонент и т. д.
	Вариант 24
1.	Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*, делящиеся на 3 и не делящиеся на 7.
2.	Дан файл *f*, компоненты которого являются целыми числами. Число компонент файла делится на 10. Записать в файл *g* наибольшее значение первых десяти компонент файла *f*, затем – следующих десяти компонент и т. д.
	Вариант 25
1.	Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*, являющиеся точными квадратами.
2.	Дан файл *f*, компоненты которого являются целыми числами. Никакая компонента файла *f* не равна нулю. Числа в файле идут в следующем порядке: десять положительных, десять отрицательных, десять положительных, десять отрицательных и т. д. Переписать компоненты файла *f* в файл *g* так, чтобы в файле *g* числа шли в следующем порядке: пять положительных, пять отрицательных, пять положительных, пять отрицательных и так далее.
	Вариант 26
1.	Дан файл *f*, компоненты u_1, u_2, …, u_n которого являются последовательными числами Фибоначчи (последовательность чисел Фибоначчи u_1, u_2, …, u_n образуется по закону $u_1 = 1$, $u_2 = 1$, $u_i = u_{i-1} + u_{i-2}$ (i = 3, 4, …). Получить в файле *f*

	последовательные числа Фибоначчи u_1, u_2, …, u_{n+1}.
2.	Дан файл *f*, компоненты которого являются целыми числами. Никакая компонента файла *f* не равна нулю. Числа в файле идут в следующем порядке: десять положительных, десять отрицательных, десять положительных, десять отрицательных и т. д. Переписать компоненты файла *f* в файл *g* так, чтобы в файле *g* числа шли в следующем порядке: двадцать положительных, двадцать отрицательных, двадцать положительных, двадцать отрицательных и т. д. (предполагается, что число компонент файла *f* делится на 40).
	Вариант 27
1.	Дан символьный файл *f*. Получить файл *g*, образованный из файла *f* заменой всех его прописных (заглавных) букв русского алфавита одноименными строчными (малыми).
2.	Дан файл *f*, компоненты которого являются целыми числами. Никакая из компонент файла не равна нулю. Файл *f* содержит столько же отрицательных чисел, сколько и положительных. Переписать компоненты файла *f* в файл *g* так, чтобы в файле g не было двух соседних чисел с одним знаком.
	Вариант 28
1.	Вычислить по схеме Горнера значение многочлена с рациональными коэффициентами для данного рационального значения переменной. Считать, что числители и знаменатели коэффициентов записаны в файле *f*: вначале числитель и знаменатель старшего коэффициента и так далее, в последнюю очередь – числитель и знаменатель свободного члена.
2.	Дан файл *f*, компоненты которого являются целыми числами. Никакая из компонент файла не равна нулю. Файл *f* содержит столько же отрицательных чисел, сколько и положительных. Переписать компоненты файла *f* в файл *g* так, чтобы в файле *g* сначала шли положительные, потом отрицательные числа.
	Вариант 29
1.	Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g* все четные числа файла *f*, а в файл *h* – все нечетные. Порядок следования чисел сохраняется.
2.	Дан файл *f*, компоненты которого являются целыми числами. Никакая из компонент файла не равна нулю. Файл *f* содержит столько же отрицательных чисел, сколько и положительных. Число компонент в файле *f* делится на 4. Переписать компоненты файла *f* в файл *g* так, чтобы в файле *g* числа шли в следующем порядке: два положительных, два отрицательных, два положительных, два отрицательных и так далее.
	Вариант 30
1.	Даны символьные файлы *f* и *g*. Записать в файл *h* сначала компоненты файла *f*, затем – компоненты файла *g* с сохранением порядка.
2.	Дан файл *f*, компоненты которого являются целыми числами. Получить файл *g*, образованный из файла *f* исключением повторных вхождений одного и того же числа.

Решение типового варианта

Рассмотрим пример оформления работы.

Пусть вариант студента 31.

	Вариант № 31.
1.	Текстовый файл. Дано положительное число ε. Последовательность a_1, a_2, a_3, … образована по следующему закону: $a_i = \left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right)\cdot\ldots\cdot\left(1-\frac{1}{i+1}\right)$ Написать программу заполнения вещественного файла f значениями a_1, a_2, a_3, … a_n получившегося ряда, остановившись на a_n, для которого выполнено условие $\lvert a_n - a_{n-1}\rvert < \varepsilon$.
2.	Типизированный файл. Компоненты файла АССОРТИМЕНТ содержат информацию об ассортименте в отделе игрушек в виде: название (например: кукла, конструктор, кубики и т. д.), возрастной диапазон (младший / старший), цена в копейках (например, 1480 и так далее). Написать программу, находящую название игрушек, стоимость которых не превышает 20 р. и которые подходят детям младшего возраста.

Сформируем отчет к лабораторной работе 9.

1. Титульный лист.

2. Сформулируем задание варианта 31.

1. Текстовый файл. Дано положительное число ε. Последовательность a_1, a_2, a_3, … образована по следующему закону:

$$a_i = \left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right)\cdot\ldots\cdot\left(1-\frac{1}{i+1}\right).$$

Написать программу заполнения вещественного файла f значениями a_1, a_2, a_3, … a_n получившегося ряда, остановившись на a_n, для которого выполнено условие $\lvert a_n - a_{n-1}\rvert < \varepsilon$.

2. Типизированный файл. Компоненты файла АССОРТИМЕНТ содержат информацию об ассортименте в отделе игрушек в виде: название (например: кукла, конструктор, кубики и т. д.), возрастной диапазон (младший/ старший), цена в копейках (например, 1480 и так далее). Написать программу, находящую название игрушек, стоимость которых не превышает 20 р. и которые подходят детям младшего возраста,

3. Напишем программу.

```
{Текстовый файл}
uses crt;
var f: text;
    a,a1,eps:real;
```

```
    i:integer;
begin
  assign(f,'proba.pas');
  write('Введите eps:');
  readln(eps);
  rewrite(f);
  a1:=(1-1/2);
  writeln(f,a1:8:4);
  i:=2;
  repeat
    a:=a1;
    a1:=a*(1-1/(i+1));
    writeln(f,a1:8:4);
    i:=i+1
  until abs(a-a1)<eps;
  close(f)
end.}
{Типизированный файл}
uses crt;
var  a,nazv,diap,cas,i:integer;
     f: file of integer;
procedure slova(a,nazv,diap,cas:integer);
begin
    write(a:3);
    case nazv of
      1:write('1. Кукла ':20);
      2:write('2. Конструктор ':20);
      3:write('3. Кубики ':20);
      4:write('4. Машина' :20);
    end;
    case diap of
      1: write('1. Младший ':20);
      2: write('2. Старший ':20);
```

```
    end;
    writeln(cas:10);
end;
begin
  randomize;
  clrscr;
  writeln('Начинаем заполнение файла случайными значениями');
  writeln;
  assign(f,'proba.pas');
  rewrite(f);
  write('№ игрушки:');
  write('Название игрушки:');
  write('Возрастной диапазон:');
  writeln('Цена:');
  for i:=1 to 60 do write('-');
  writeln;
  for i:=1 to 15 do begin
    a:=i;
    nazv:=random(3)+1;
    diap:=random(2)+1;
    cas:=random(2000);
    slova(a,nazv,diap,cas);
    write(f, a,nazv,diap,cas);
  end;
  close(f);
  readkey;
  writeln('Задание');
  writeln('Игрушки, стоимость которых не превышает 20 руб');
  writeln('и которые подходят детям младшего воз-раста');
  reset(f);
```

```
  for i:=1 to 15 do begin
    read(f, a,nazv,diap,cas);
    if (cas<2000) and (diap=1) then
    slova(a,nazv,diap,cas);
  end;
  readkey;
end.
```

4. В текстовом редакторе WORD или рукописно оформим отчет в соответствии с содержанием.

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1. Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Абрамов, С. А. Начала информатики / С. А. Абрамов, Е. В. Зима. – М. : Наука, 1989. – 256 с.

3. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

4. Довгаль, С. И. Персональные ЭВМ : ТурбоПаскаль V 7.0. Объектное программирование. Локальные сети : учебное пособие / С. И. Довгаль, Б. Ю. Литвинов, А. И. Сбитнев.

5. Новичков, В. С. Паскаль : учеб. пособие для сред. спец. учеб. заведений / В. С. Новичков, Н. И. Парфилова, А. Н. Пылькин. – М. : Высш. шк., 1990. – 223 с. : ил. – (Алгоритмические языки в техникуме).

6. Хершель, Рудольф. Турбо Паскаль / Рудольф Хершель. – 2-е изд., перераб., – Вологда : МП «МИК», 1991. – 342 с. при участии МП ТПО «Квадрат», г. Москва.

7. Эрбс, Х.-Э. Введение в программирование на языке Паскаль : пер. с нем. / Х.-Э. Эрбс, О. Штольц. – М. : Мир, 1989. – 299 с., ил.

8. Григас, Г. Начала программирования : кн. для учащихся : пер. с лит. / Г. Григас ; под. ред. Ю. А. Первина.– М. : Просвещение, 1987. – 112 с. : ил.

9. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

Вопросы для самопроверки

1. Дайте определение файла. Укажите три разновидности файлов в зависимости от организации в них данных.

2. Укажите процедуру для задания связи конкретного текстового файла на диске с файловой переменной. Укажите процедуру закрытия доступа к файлам.

3. Укажите процедуры для открытия текстового файла на запись (создание или очистка файла, изменение файла). Укажите процедуру записи данных в текстовый файл. Опишите формат всех этих процедур.

4. Укажите процедуру для открытия текстового файла на чтение. Укажите процедуру чтения данных из текстового файла. Опишите формат обеих процедур.

5. Укажите функции анализа данных в текстовом файле (конец строки, конец файла). Для каждого служебного слова укажите формат и действие.

6. Укажите процедуру для задания связи конкретного типизированного файла на диске с файловой переменной. Укажите процедуру закрытия доступа к файлам.

7. Укажите процедуры, которыми можно открыть типизированный файл на запись данных (создание или очистка файла, изменение файла). Укажите процедуру записи данных в типизированный файл. Опишите формат всех этих процедур.

8. Укажите процедуры, которыми можно открыть типизированный файл для чтения данных. Укажите процедуру чтения данных из типизированного файла. Опишите формат всех этих процедур.

9. Укажите функции анализа данных в типизированном файле (конец файла, позиция указателя в файле, количество компонент). Для каждого служебного слова укажите формат и действие.

10. Укажите служебные слова, которые называются операторными скобками. Определите ситуации и операторы, для которых необходимы операторные скобки. Приведите пример.

ЛАБОРАТОРНАЯ РАБОТА 10.
СТРУКТУРИРОВАННЫЕ ТИПЫ. ЗАПИСЬ И МНОЖЕСТВО (2 ЧАСА)

Цель работы: освоить оставшиеся два из четырех стандартных структурированных типов Турбо Паскаля. Узнать способы использования поля в типе ЗАПИСЬ и конструктора в типе МНОЖЕСТВО. Исследовать примеры на тему «Запись и множество».

Теоретические положения
Записи

Запись – это структура данных, состоящая из фиксированного числа компонентов, называемых полями записи. В отличие от массива, компоненты (поля) записи могут быть различного типа. Чтобы можно было ссылаться на тот или иной компонент записи, поля именуются.

Структура объявления типа ЗАПИСЬ такова:

<имя типа> = **RECORD** <сп.полей> **END**

Здесь

<имя типа> – правильный идентификатор;

record – зарезервированное слово (запись);

[<сп.полей>] – список полей, представляет собой последовательность разделов записи, между которыми ставится точка с запятой.

end – зарезервированное слово (конец).

Каждый раздел записи состоит из одного или нескольких идентификаторов полей, отделяемых друг от друга запятыми. За идентификатором (идентификаторами) ставится двоеточие и описание типа поля (полей)

Пример.

```
type
  Birthday = record
        day,month: byte;

        year          : word
        end;
var
  a, b: Birthday;
```

В этом примере тип Birthday (день рождения) содержит три поля с именами day, month и year (день, месяц и год); переменные *a* и *b* содержат записи типа Birthday.

Как и в массиве, значения переменных типа записи можно присваивать другим переменным того же типа, например:

```
a := b;
```

К каждому из компонентов записи можно получить доступ, если использовать составное имя, то есть указать имя переменной, затем точку и имя поля:

```
a.day := 17;
b.year := 2008;
```

Для вложенных полей приходится продолжать уточнения:

```
var
  c: record
        name: string;
        bd: Birthday
        end;
```

Тогда в программе можно использовать оператор

```
if c.bd.year = 2008 then ...
```

Пример. Составить программу для зачисления на стипендию студентов групп A, B, C, состоящих соответственно из 20, 24 и 23 человек, используя подпрограмму общего вида.

```
type
  student = record
    name: string;
```

```
    oc: array[1..3] of byte;
    step: boolean
    end;
  mas = array[1..24] of student;
var
  a, b, c: mas;
  i, j: byte;
procedure Wwod(var a: mas; n: byte);
var
  i, j: byte;
Begin
  for i := 1 to n do begin
    writeln('Студент № ', i);
    write('Ф.И.О.'); readln(a[i].name);
    a[i].step := true;
    for j:=1 to 3 do begin
      write(j, ' – '); readln(a[i].oc[j]);
      if a[i].oc[j] < 3 then a[i].step := false;
      end;
    end;
End;
procedure Wywod(a: mas; n: byte);
var i: byte;
Begin
  for i:=1 to n do
    if a[i].step then writeln(a[i].name)
End;
BEGIN
  Wwod(a, 20);
  Wwod(b, 24);
  Wwod(c, 23);
  Wywod(a, 20);
  Wywod(b, 24);
  Wywod(c, 23);
END.
```

Пример. Составить программу выстраивания фамилий списка студентов в порядке их рейтинга.

```
type
  student = record
    name: string;
    numer: byte;
    rang: byte
    end;
  mas = array[1..4] of student;
var
  a: mas;
  i, j: byte;
procedure Wwod(var a: mas; n: byte);
var i, j: byte;
Begin
  for i:=1 to n do begin
    writeln('Студент № ', i);
    a[i].numer := i;
    write('Ф.И.О.'); readln(a[i].name);
    write('рейтинг – '); readln(a[i].rang);
    end;
End;
procedure Rang_Gr(var a: mas; n: byte);
var i, j, k: byte;
pr: student;
Begin
  for j := 1 to n do begin
    k := j;
    for i := j + 1 to n do
      if a[i].rang < a[k].rang then k := i;
    pr := a[i];
    a[i] := a[k];
```

```
    a[k] := pr
  end
End;
procedure Wywod(a: mas; n: byte);
var i: byte;
Begin
  for i := 1 to n do
    writeln(a[i].name, a[i].rang, a[i].numer)
End;
BEGIN
  Wwod(a, 4);
  Rang_Gr(a, 4);
  Wywod(a, 4)
END.
```

Чтобы упростить доступ к полям записи, используется оператор присоединения **WITH**:

WITH <переменная> **DO** <оператор>

Здесь

with – зарезервированное слово (с);

<переменная> – имя переменной типа запись, за которым, возможно, следует список вложенных полей.;

do – зарезервированное слово (делать);

<оператор> – любой оператор Турбо Паскаля.

Например:

```
with c.bd do month := 9;
```

Это равносильно

```
c.bd.month := 9;
```

Понятно, что использование оператора **with** оправдано для большого количества действий с записями.

Вывод на экран значений переменной типа запись возможно только для каждого поля по отдельности.

Пример. Написать программу заполнения ведомости выплат случайными значениями.

```
uses crt;
type
  proba = record
    name: string;
    cas: word
    end;
var a, b: proba;
begin
  clrscr;
  randomize;
  a.name := 'Маша';
  a.cas := random(10000);
  b.name := 'Саша';
  b.cas := random(10000);
  writeln(a.name:6, a.cas:8);
  writeln(b.name:6, b.cas:8);
  readkey
end.
```

Множества

Множества – это наборы однотипных логически связанных друг с другом объектов. Характер связей лишь подразумевается программистом и никак не контролируется Турбо Паскалем. Количество элементов, входящих в множество, может меняться в пределах от 0 до 256 (множество, не содержащее элементов, называется пустым). Именно непостоянством количества своих элементов множества отличаются от массивов и записей.

Два множества считаются эквивалентными тогда и только тогда, когда все их элементы одинаковы, причем порядок следования элементов безразличен. Если все элементы одного множества входят

также и в другое, говорят о включении первого множества во второе. Пустое множество включается в любое другое.

Пример определения и задания множеств:

```
type
  digitChar = set of ‘0’..’9’;
  digit     = set of 0..9;
var
  s1, s2, s3: digitChar;
  s4, s5, s6: digit;
.......
s1 := [‘1’, ’2’, ’3’];
s2 := [’3’, ’2’, ’1’];
s3 := [’2’, ’3’];
s4 := [0..3, 6];
s5 := [4, 5];
s6 :=[4..9];
```

В этом примере множества s1 и s2 эквивалентны, а множество s3 включено в s2, но не эквивалентно ему.

Описание типа множества имеет вид:

<имя типа> = **SET OF** <баз.тип>

Здесь

<имя типа> – правильный идентификатор;

set – зарезервированное слово (множество)

of – зарезервированное слово (из)

<баз.тип> – базовый тип элементов множества, в качестве которого может использоваться любой порядковый тип, кроме word, integer, longint.

Для задания множества используется так называемый **конструктор множества**: список спецификаций элементов множества, отделяемых друг от друга запятыми, список обрамляется квадратны-

ми скобками. Спецификациями элементов могут быть константы или выражения базового типа, а также – тип-диапазон того же базового типа.

Над множествами определены следующие операции:

* – пересечение множеств, результат содержит элементы, общие для обоих множеств, например:

s4 * s6 содержит [6],

s4 * s5 содержит [] (пустое множество),

+ – объединение множеств, результат содержит элементы первого множества, дополненные недостающими элементами из второго множества.

s4 + s5 содержит [0, 1, 2, 3, 4, 5, 6],

s5 + s6 содержит 4, 5, 6, 7, 8, 9],

– – разность множеств, результат содержит элементы из первого множества, которые не принадлежат второму.

s6 – s5 содержит [6, 7, 8, 9],

s4 – s5 содержит [0, 1, 2, 3, 6],

= – проверка эквивалентности, возвращает TRUE, если множества эквивалентны.

<> – проверка неэквивалентности, возвращает TRUE, если множества неэквивалентны.

<= – проверка вхождения, возвращает TRUE, если первое множество включено во второе.

>= – проверка вхождения, возвращает TRUE, если второе множество включено в первое.

IN – проверка принадлежности, в этой бинарной операции первый элемент – выражение, а второй – множество одного и того же типа, возвращает TRUE, если выражение имеет значение, принадлежащее множеству.

4 **in** s6 возвращает TRUE,

2 * 2 **in** s1 возвращает FALSE.

Пример.

Выделить из первой сотни натуральных чисел все простые числа.

```
const n = 100;
type natur = set of 1..n;
```

```
var
  n1, next, i: word;
  beginSet, endSet: natur;
begin
  beginSet:= [2..n];
  endSet:= [1];
  next:= 2;
  while beginSet <> [] do begin
    n1 := next;
    while n1 <= n do begin
      beginSet := beginSet – [n1];
      n1 := n1 + next
      end;
    endSet := endSet + [next];
    repeat
      inc(next)
    until (next in beginSet) or (next > n)
  end;
  for i := 1 to n do
    if i in endSet then write(i: 8);
  writeln
  end.
```

Пример. Выяснить, какие различные числа содержит поток данных. В потоке данных встречаются числа от 0 до 255.

```
type short=set of 0..255;
var
  potok: short;
  sh: integer;
  i: byte;
begin
  potok := [];
  writeln('Введи данные');
  repeat
    readln(sh);
    potok := potok + [sh];
```

```
  until (sh < 0) or (sh > 255);
  potok := potok – [sh];
  writeln('Вывод значений потока');
  for i := 0 to 255 do
    if i in potok then writeln(i)
end.
```

Порядок выполнения работы

1. Уточните номер своего варианта.
2. Напишите программу.
3. Отладьте программу (исключите все сообщения об ошибках).
4. В текстовом редакторе WORD или рукописно оформите отчет в соответствии с содержанием.

Содержание отчета

1. Титульный лист.
2. Задания варианта
3. Программные единицы.

Варианты задач

	Вариант 1
1.	Создать список работников предприятия. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, пол, год рождения, семейное положение. 2. Вывести массив в табличном виде. 3. Вывести все сведения об Ивановых, затем найти количество отделов на предприятии.
2.	Сформировать множества **A** и **B**, базовый тип которых *-10..30*, по заданному числу элементов для каждого множества. Определить, какое из этих множеств содержит наибольший элемент, не принадлежащий другому множеству, затем найти число и сумму элементов, принадлежащих одновременно **A** и **B**.
	Вариант 2
1.	Создать список жильцов дома. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, № квартиры, № этажа, число членов семьи, право на льготы. 2. Вывести массив в табличном виде. 3. Вывести списки жильцов четвертого этажа, затем число жильцов, имеющих льготы.

2.	Сформировать множества **A** и **B**, базовый тип которых *10..60*, по заданному числу элементов для каждого множества. Определить, в каком из множеств больше элементов, не принадлежащих другому множеству. Удалить из **A** все элементы, не принадлежащие **B.**
	Вариант 3
1.	Создать список школьников. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, год рождения, фамилия родителя, вес, рост, пол, поведение (уд./неуд). 2. Вывести массив в табличном виде. 3. Вывести списки школьников младше 12 лет, затем самого высокого ребенка.
2.	Сформировать множества **A** и **B**, базовый тип которых *-20..20*, по заданному числу элементов для каждого множества. Найти сумму элементов **A**, не принадлежащих **B,** и число элементов, принадлежащих и тому и другому множеству.
	Вариант 4
1.	Создать список студентов группы. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, возраст, номер студенческого билета, оценки за последнюю сессию (по трем предметам). 2. Вывести массив в табличном виде. 3. Вывести списки тех, у кого средний балл выше 4, затем студентов – моложе 20 лет.
2.	Сформировать множества **A** и **B**, базовый тип которых *20..50*, по заданному числу элементов для каждого множества. Определить, в каком из множеств меньше элементов, затем найти среднее арифметическое элементов каждого из множеств.
	Вариант 5
1.	Создать список клиентов банка. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, номер счета, сумма на счету (руб.), пенсионный ли счет (да/нет). 2. Вывести массив в табличном виде. 3. Вывести списки клиентов со счетом > 10 тыс. рублей, затем клиентов, имеющих особый годовой процент начисления как пенсионный.
2.	Сформировать множества **A, B** и **C**, базовый тип которых *–50.. –10*, по заданному числу элементов для каждого множества. Подсчитать процентное количество числа элементов каждого из них по отношению к общему числу элементов базового типа; и процентное количество элементов базового множества, не принадлежащих ни одному из них.
	Вариант 6
1.	Создать список продуктовых товаров. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название (например: макароны, мясо, полуфабрикаты и так далее), дата изготовления, срок годности, фирма-изготовитель, вес Нетто, требование особого

	температурного режима хранения (да/нет). 2. Вывести массив в табличном виде. 3. Вывести списки товара, годного до июня 2009 года, затем изготовленного фирмой Макфа.
2.	Сформировать множества **A, B** и **C**, базовый тип которых *20..60*, по заданному числу элементов для каждого множества. Определить, является ли одно из них подмножеством другого множества, затем определить, в каком из множеств больше элементов.
	Вариант 7
1.	Создать список спортсменов. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, имя, возраст, вес, рост, город. 2. Вывести массив в табличном виде. 3. Вывести списки спортсменов 1986 года рождения, чей вес превышает 50 кг, затем найти самого высокого спортсмена.
2.	Сформировать множества **A** и **B**, базовый тип которых *–70.. –30*, по заданному числу элементов для каждого множества. Найти наибольший элемент, принадлежащий одновременно **A** и **B**, затем количество элементов, входящих в **A** и не входящих в **B.**
	Вариант 8
1.	Создать список книг домашней библиотеки. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название, автор, номер учета, год издания, издательство, возрастные ограничения (детская – да/нет). 2. Вывести массив в табличном виде. 3. Вывести списки книг, изданных раньше 1990 г., затем найти самую старую книгу из списка.
2.	Сформировать множества **A, B** и **C**, базовый тип которых *20..60*, по заданному числу элементов для каждого множества. Найти среднее арифметическое элементов каждого из множеств и вывести на печать элементы, входящие одновременно во все эти множества.
	Вариант 9
1.	Создать список призывников. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, возраст, рост, адрес, место работы / учебы, право на отсрочку, причина (болезнь, учеба или семейное положение). 2. Вывести массив в табличном виде. 3. Вывести списки тех, которые моложе 18 лет, затем найти средний рост призывников.
2.	Сформировать множества **A, B** и **C**, базовый тип которых *–100..–60*, по заданному числу элементов для каждого множества. Определить, во множестве **A** или **B** больше элементов, принадлежащих **C**. Найти число элементов множества **C**, без элементов, которые принадлежат **A** или **B.**

	Вариант 10
1.	Создать список картин выставки иностранных мастеров. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия автора, год его рождения, название картины. 2. Вывести массив в табличном виде. 3. Найти общую сумму лет жизни, представленных в галерее художников, затем вывести список произведений художника Моне.
2.	Сформировать множества **A** и **B**, базовый тип которых *10..50*, по заданному числу элементов для каждого множества. Определить, в каком из множеств больше элементов. Найти среднее арифметическое элементов каждого из множеств и количество элементов базового типа, не входящих ни в **A**, ни в **B.**
	Вариант 11
1.	Создать список работников фирмы. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, пол, возраст, адрес (улица, номер дома и квартиры), название отдела. 2. Вывести массив в табличном виде. 3. Вывести списки работников не старше 45 лет, затем найти количество работников отдела, название которого введено с клавиатуры.
2.	Сформировать множества **A** и **B**, базовый тип которых *–30..10*, по заданному числу элементов для каждого множества. Сформировать множество **C** из тех элементов **A** и **B**, которые не принадлежат **A** и **B** одновременно. Найти число элементов в полученных трех множествах.
	Вариант 12
1.	Создать список имеющейся на торговой базе краски. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: цвет, месяц и год изготовления, состав (масляная, синтетическая или водная), пригодность применения в детских учреждениях (да/нет) 2. Вывести массив в табличном виде. 3. Вывести записи по масляной краске, затем краски, изготовленные позже 01.06.2007.
2.	Сформировать множества **A, B, C**, базовый тип которых *50..90*, по заданному числу элементов для каждого множества. Найти среднее арифметическое каждого из множеств, затем вывести на печать элементы, входящие одновременно во все множества.
	Вариант 13
1.	Создать список учащихся колледжа. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, имя, пол, год рождения, номер визитки, средний балл за предыдущий год. 2. Вывести массив в табличном виде. 3. Вывести списки учащихся, упорядоченные по возрасту, затем найти учащегося, визитка которого зарегистрирована позже других.
2.	Сформировать множества **A** и **B**, базовый тип которых *–60..–20*, по заданному числу элементов для каждого множества. Определить множество, содер-

	жащее наименьший элемент, принадлежащий другому множеству, затем найти число и сумму элементов, принадлежащих одновременно и **A**, и **B**.
	Вариант 14
1.	Создать список абонентов телефонной сети. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, имя, адрес (улица, номер дома и квартира), номер телефона, вид оплаты (абонентская/поминутная). 2. Вывести массив в табличном виде. 3. Вывести номер телефонов всех абонентов Ивановых, затем абонентов по ул. Краматорской.
2.	Сформировать множества **A**, **B** и **C**, базовый тип которых *70..110*, по заданному числу элементов для каждого множества. Определить, в каком из множеств больше элементов, не принадлежащих другому, затем убрать из **A** все элементы, принадлежащие одновременно **B**.
	Вариант 15
1.	Создать список детей ясельной группы детского сада. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, имя, год рождения, домашний телефон, эмблема на шкафчике, наличие льгот на оплату (да/нет). 2. Вывести массив в табличном виде. 3. Вывести списки малышей с одинаковой эмблемой, затем льготников, достигших трех лет.
2.	Сформировать множества **A** и **B**, базовый тип которых *–60..–20*, по заданному числу элементов для каждого множества. Определить, является ли одно из них подмножеством другого, и найти число элементов, принадлежащих разности большего и меньшего из этих множеств.
	Вариант 16
1.	Создать список ассортимента в посудной лавке. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название посуды, количество экземпляров, срок использования (одноразовая/обыкновенная), материал (стекло, керамика, пластмасса и так далее), цена. 2. Вывести массив в табличном виде. 3. Вывести списки посуды с одинаковым названием, затем найти самый дешевый способ накрыть стол на шесть человек (то есть три блюда по шесть комплектов).
2.	Сформировать множества **A** и **B**, базовый тип которых *10..50*, по заданному числу элементов для каждого множества. Подсчитать процентное количество числа элементов каждого из них по отношению к общему числу элементов базового типа и процентное количество элементов базового множества, не принадлежащих ни одному их них.
	Вариант 17
1.	Создать список имеющихся в аптеке лекарственных трав. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название, дата изготовления (день, месяц и год), вес, цена, срок годности, сбор (промышленный/любительский). 2. Вывести массив в табличном виде. 3. Вывести записи по травам, изготовленным в 2001, упорядоченные по цене,

	затем найти количество трав с одинаковым сроком годности.
2.	Сформировать множества **A** и **B**, базовый тип которых *-30..10*, по заданному числу элементов для каждого множества. Найти сумму элементов **A**, не принадлежащих **B**, затем произведение элементов **B**, не принадлежащих **A.**
	Вариант 18
1.	Создать список легкоатлетов. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, возраст, вес, рост, звание (кандидат в мастера спорта, мастер), профиль (профессионал/любитель). 2. Вывести массив в табличном виде. 3. Вывести списки спортсменов-прыгунов, упорядоченные по росту, найти данные самого пожилого легкоатлета.
2.	Сформировать множества **A** и **B**, базовый тип которых *110..150*, по заданному числу элементов для каждого множества. Найти число и произведение элементов, принадлежащих одновременно **A** и **B.**
	Вариант 19
1.	Создать список родственников. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, имя, пол, возраст, кем приходится, дата рождения. 2. Вывести массив в табличном виде. 3. Вывести списки родственников в алфавитном порядке, затем найти самого молодого родственника
2.	Сформировать множества **A, B** и **C**, базовый тип которых *–50..–10*, по заданному числу элементов для каждого множества. Определить множество, содержащее наименьший элемент, принадлежащий другому множеству.
	Вариант 20
1.	Создать список учета сантехники на торговой базе. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название (например: кран, смеситель, унитаз и так далее), страна-изготовитель (РФ, Германия, Белоруссия и так далее), номер учета, год изготовления. 2. Вывести массив в табличном виде. 3. Вывести список сантехники, изготовленной раньше введенной с клавиатуры даты, затем вывести список устройств, изготовленных в Германии.
2.	Сформировать множества **A**, **B** и **C**, базовый тип которых *60..110*, по заданному числу элементов для каждого множества. Найти среднее арифметическое элементов каждого из множеств и всех множеств.
	Вариант 21
1.	Создать список учета пассажиров в поезде. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, номер места в вагоне, место прибытия, категория (купе/плацкарт), количество вещей и общий вес вещей. 2. Вывести массив в табличном виде. 3. Дать сведения о трех пассажирах, число вещей в багаже которых меньше, чем в любом другом багаже, затем вычислить общее количество вещей в списке.

2.	Сформировать множества **A** и **B**, базовый тип которых *-20..20*, по заданному числу элементов для каждого множества. Найти первый по возрастанию элемент **A**, не принадлежащий **B,** затем найти число совпадающих в них элементов.
	Вариант 22
1.	Создать список видеофильмов проката. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название фильма, год создания, жанр произведения, имя режиссера, разрешен ли показ детям до 16 лет (да/нет). 2. Вывести массив в табличном виде. 3. Вывести списки имеющихся фильмов конкретного режиссера, имя которого введено с клавиатуры.
2.	Сформировать множества **A** и **B**, базовый тип которых *10..60*, по заданному числу элементов для каждого множества. Удалить из **A** все элементы, большие среднего арифметического четных элементов **A**. Убрать из **B** все элементы, большие среднего арифметического нечетных элементов **B.**
	Вариант 23
1.	Создать список сотрудников учреждения. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, имя, отчество, пол, дата рождения, стаж работы. 2. Вывести массив в табличном виде. 3. Вывести на экран данные о сотрудниках, юбилей которых будет праздноваться в текущем году (юбилей для мужчин – возраст или стаж кратный 10, для женщин – стаж, кратный 5).
2.	Сформировать множества **A** и **B**, базовый тип которых *–80. –40*, по заданному числу элементов для каждого множества. Удалить из **B** все элементы, меньшие среднего арифметического элементов множества **A**, затем определить, в каком из множеств больше элементов.
	Вариант 24
1.	Создать список сведений о детских кубиках. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: размер (длина ребра в миллиметрах), цвет (красный, желтый, зеленый или какой-либо другой) и материал (деревянный, металлический, картонный или какой-либо другой). 2. Вывести массив в табличном виде. 3. Вывести количество кубиков каждого из перечисленных цветов, затем объемы израсходованных материалов.
2.	Сформировать множество **A**, базовый тип которого *50..90*, по заданному числу элементов для каждого множества. Найти число элементов множества **A** между наибольшим и наименьшим из элементов **A.**
	Вариант 25
1.	Создать список экземпляров животного мира в зоосаде. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название, регион обитания в природе, возраст, высота в холке, опасность для человека (да/нет).

	2. Вывести массив в табличном виде. 3. Вывести на экран данные о трех самых младших обитателях зоосада, затем вывести список опасных.
2.	Сформировать множества **A, B** и **C**, базовый тип которых *–120.. –80*, по заданному числу элементов для каждого множества. Найти количество одинаковых элементов для **A** и **B, B** и **C, C** и **A**.
	Вариант 26
1.	Создать список международной системы единиц. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: величина (длина, масса, время и так далее), наименование единицы (метр, килограмм, секунда и так далее), обозначение международное (m, kg, s и так далее), обозначение русское (м, кг, с и так далее), размер единицы (определен международным соглашением, м*м, кг/(м*м*м) и так далее). 2. Вывести массив в табличном виде. 3. Вывести на экран данные об основных единицах, затем упорядочить все сведения по алфавиту условных международных обозначений.
2.	Сформировать множества **A, B** и **C**, базовый тип которых *30..70*, по заданному числу элементов для каждого множества. Определить, в каком из множеств больше элементов, и вывести на печать те элементы базового интервала, которые не входят в него.
	Вариант 27
1.	Создать список стран политической карты мира. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: официальное название страны, столица, общая площадь и население, является ли членом Совета Безопасности ООН (да/нет). 2. Вывести массив в табличном виде. 3. Дать сведения о странах, являющихся членом Совета Безопасности ООН, затем о странах, общая площадь которых меньше 100 тыс. кв. км, а население больше введенного с клавиатуры значения.
2.	Сформировать множества **A** и **B**, базовый тип которых *–20..20*, по заданному числу элементов для каждого множества. Найти сумму нечетных элементов, принадлежащих и тому, и другому из множеств. Определить в каком множестве больше элементов, не принадлежащих другому множеству.
	Вариант 28
1.	Создать информационный список известных людей России. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: фамилия, имя, пол, род деятельности (наука, управление, торговля и так далее), дата рождения, жив или нет. 2. Вывести массив в табличном виде. 3. Вывести списки дореволюционных деятелей науки, затем найти персоналии с отчеством Михайлов (ич/на).
2.	Сформировать множества **A, B** и **C**, базовый тип которых *40..80*, по заданному числу элементов для каждого множества. Найти одинаковые элементы для **A** и **B, B** и **C, C** и **A,** а также для **A, B и C**.
	Вариант 29

1.	Создать справочник языка программирования, который изучаете. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: слово (if, record, read и так далее), значение (если, запись, читать и так далее), является ли служебным(да/нет), является ли предопределенным идентификатором (да/нет). 2. Вывести массив в табличном виде. 3. Вывести списки служебных слов по алфавиту, затем найти все предопределенные идентификаторы, начинающиеся на букву **W**.
2.	Сформировать множества **A** и **B**, базовый тип которых *–70.. –30*, по заданному числу элементов для каждого множества. Найти сумму элементов множества **C**, которое содержит каждый элемент множества **A**, не принадлежащий **B**.
	Вариант 30
1.	Создать информационный список, содержащий сведения об игрушках. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название (например: кукла, конструктор, кубики и так далее), возрастной диапазон (младший/старший), цена в копейках не более 20 руб. (например, 14480 и т. д.). 2. Вывести массив в табличном виде. 3. Вывести запись о самом дорогом конструкторе, затем получить названия наиболее дорогих игрушек (цена которых отличается от цены самой дорогой игрушки не более чем на 1 руб.).
2.	Сформировать множества **A** и **B**, базовый тип которых *20..60*, по заданному числу элементов для каждого множества. Найти сумму элементов, принадлежащих тому и другому множеству, затем определить, в каком множестве больше элементов.

Решение типового варианта

Рассмотрим пример оформления работы.

Пусть вариант студента 31.

	Вариант 31
1.	Создать информационный список, содержащий сведения об игрушках. 1. Описать и ввести массив, содержащий 10 записей следующей структуры: название (например: кукла, конструктор, кубики и т. д.), возрастной диапазон (младший возраст/старший), цена в копейках (например, 25480 и т. д.). 2. Вывести массив в табличном виде. 3. Вывести на экран название игрушек, стоимость которых не превышает введенную с клавиатуры сумму и которые подходят детям младшего возраста, затем стоимость самого дорогого конструктора.
2.	Сформировать множества **A** и **B**, базовый тип которых – прописные буквы латинского алфавита ‘A’..’Z’, по заданному числу элементов для каждого множества. Найти число элементов, принадлежащих одновременно **A** и **B**; выведите на экран все элементы множества **A**, не принадлежащие **B**.

Сформируем отчет к лабораторной работе 10 .

1. Титульный лист.

2. Сформулируем задание варианта 31.

1. Запись.

Создать информационный список, содержащий сведения об игрушках.

1. Описать и ввести массив, содержащий 10 записей следующей структуры: название (например: кукла, конструктор, кубики и т. д.), возрастной диапазон (младший возраст/старший), цена в копейках (например, 25480 и т. д.).

2. Вывести массив в табличном виде.

3. Вывести на экран название игрушек, стоимость которых не превышает введенную с клавиатуры сумму и которые подходят детям младшего возраста, затем стоимость самого дорогого конструктора.

2. Множество.

Сформировать множества **A** и **B**, базовый тип которых – прописные буквы латинского алфавита ‘A’..’Z’, по заданному числу элементов для каждого множества. Найти число элементов, принадлежащих одновременно **A** и **B**; выведите на экран все элементы множества **A**, не принадлежащие **B**.

4. Напишем программу.

```
1.
uses crt;
type
  assort=record
    name:string[20];
    vozr:boolean;
    cas:integer;
    end;
  assorty=array[1..10] of assort;
var
  i:byte;
  c:integer;
  max:assort;
const
```

```
  a:assorty=((name:'кукла'; vozr:false; cas:200),
        (name:'конструктор'; vozr:true; cas:1420),
           (name:'кубики'; vozr:true; cas:1000),
           (name:'кукла'; vozr:false; cas:510),
           (name:'кубики'; vozr:false; cas:290),
        (name:'конструктор'; vozr:true; cas:1400),
           (name:'кукла'; vozr:false; cas:1320),
           (name:'кукла'; vozr:false; cas:1340),
        (name:'конструктор'; vozr:true; cas:2000),
           (name:'кукла'; vozr:false; cas:1540));
begin
  clrscr;
writeln('Название':10,'Возраст':10,'Цена':10,#10);
  for i:=1 to 10 do begin
    write(a[i].name:12);
    if a[i].vozr then write('старший':12)
                 else write('младший':12);
    writeln(a[i].cas:12);
  end;
  writeln('Название игрушек, которые подходят де-
тям младшего возраста');
  writeln('и не превышают введенную цену');
  writeln('Введите цену в рублях');
  readln(c);
  c:=100*c;
  for i:=1 to 10 do
    if (a[i].cas<=c) and not a[i].vozr then begin
    write(a[i].name:10);
    if a[i].vozr then write('старший':10)
                 else write('младший':10);
    writeln(a[i].cas:10);
  end;
  writeln;
```

```
  writeln('Стоимость самого дорогого конструкто-
ра');
  i:=1;
  repeat
  i:=i+1
  until a[i].name='конструктор';
  max:=a[i];
  for i:=1 to 10 do
    if (a[i].name='конструктор') and
       (max.cas<a[i].cas)then max:=a[i];
    write(max.name:10);
    if max.vozr then write('старший':10)
                else write('младший':10);
    writeln(max.cas:10);
  readkey
end.
2.
uses crt;
type
  mnog = set of char;
var
  a, b, c: mnog;
  m: set of 0..255;
  y, x, k, n: byte;
begin
  clrscr;
  randomize;
  m := [65..90]; {по кодовой таблице}
  writeln;
  writeln('введите кол-во элементов множества A');
  readln(n);
  a := [];
  repeat
    k := 0;
```

```
    x := random(255);
    if x in m then a := a + [chr(x)];
    for y := 0 to 255 do
      if chr(y) in a then k:=k+1;
  until k >= n;
  writeln('множество A');
  for x := 0 to 255 do
    if chr(x) in a then write(chr(x):5);
  writeln;
  writeln('введите кол-во элементов множества B');
  readln(n);
  b := [];
  repeat
    k := 0;
    x:=random(255);
    if x in m then b:=b+[chr(x)];
    for y := 0 to 255 do
      if chr(y) in b then k:=k+1;
  until k>=n;
  writeln('множество B');
  for x:=0 to 255 do
    if chr(x) in b then write(chr(x):5);
  writeln;
  writeln('множество C = A + B');
  c := a + b;
  k:=0;
  for x:=0 to 255 do
    if chr(x) in c then begin
      write(chr(x):5);
      k:=k+1
    end;
  writeln;
  writeln('число элементов принадлежащих A и B =
', k);
```

```
  writeln('множество C = A - B');
  c:=a-b;
  for x := 0 to 255 do
    if chr(x) in c then write(chr(x):5);
  writeln;
  readkey
end.
```

5. В текстовом редакторе WORD или рукописно оформим отчет в соответствии с содержанием.

Список литературы

1. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1 .Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

2. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

3. Довгаль, С. И. Персональные ЭВМ: ТурбоПаскаль V 7.0. Объектное программирование. Локальные сети : учебное пособие / С. И. Довгаль, Б. Ю. Литвинов, А. И. Сбитнев.

4. Епанешников, А. Программирование в среде Turbo Pascal 7.0 / А. Епанешников, В. Епанешников. – М. : «ДИАЛОГ-МИФИ», 1993. – 288 с.

5. Новичков, В. С. Паскаль : учеб. пособие для сред. спец. учеб. заведений / В. С. Новичков, Н. И. Парфилова, А. Н. Пылькин. – М. : Высш. шк., 1990. – 223 с. : ил. – (Алгоритмические языки в техникуме).

6. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

Вопросы для самопроверки

1. Дайте формат структурированного типа **RECORD** (запись). Приведите пример задания типа запись для обработки данных о дате (день, месяц, год).

2. Задайте запись **ADDRESS** с полями: **CITY** – строка символов, **HOUSE** – целое число в диапазоне от 1 до 1000, **INDEX** – по-

метка, принимающая значения ‘**Да’** (**TRUE**) или ‘**Нет’ (FALSE).** Приведите пример задания переменной **ADDRESS** номера дома.

3. Дайте формат определения структурированного типа множества. Покажите конструктор целочисленного множества для переменной *S*, задающий значения 1, 2, 5.

4. Задайте тип множества для переменной *S_CHAR* для обработки элементов типа **CHAR**.

5. Дайте определение пересечения множеств. Укажите условное обозначение пересечения множеств. Предскажите результат пересечения множеств A = {1, 2, 3} и B = {2..4}.

6. Дайте определение объединения множеств. Укажите условное обозначение пересечения множеств. Предскажите результат объединения множеств A = {1, 2, 3} и B = {2..4}.

7. Дайте определение разности множеств. Укажите условное обозначение разности множеств. Предскажите результат разности множеств A = {1, 2, 3} и B = {2..4}.

8. Дайте определение эквивалентности множеств. Укажите условное обозначение проверки эквивалентности множеств. Предскажите результат проверки эквивалентности множеств A = {1, 2, 3} и B = {1..3}.

9. Дайте определение неэквивалентности множеств. Укажите условное обозначение проверки неэквивалентности множеств. Предскажите результат проверки неэквивалентности множеств A = {1, 2, 3} и B = {1..3}.

10. Дайте определение вхождения одного множества в другое. Укажите условное обозначение проверки вхождения одного множества в другое. Предскажите результат проверки неэквивалентности множеств A = {1, 2, 3} и B = {1..3}.

11. Дайте определение принадлежности элемента множеству. Укажите условное обозначение проверки принадлежности элемента множеству. Предскажите результат проверки принадлежности элемента 3 множеству A = {1, 2, 3}.

12. Перечислите символьные типы данных. Укажите основные свойства данных, которые может хранить каждый из них и служебные слова, задающие эти типы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Методика преподавания информатики по машинному варианту (10-11 класс) : методические рекомендации / под ред. В. А. Каймина (Часть 1, 2). Московский городской комитет по народному образованию. Московский городской институт усовершенствования учителей. – Москва, 1990.

2. Фаронов, В. В. Турбо Паскаль : в 3 кн. / В. В. Фаронов. – Книга 1 .Основы Турбо Паскаля. – М. : Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992. – 304 с., ил.

3. Радер, Дж. Бейсик для персонального компьютера фирмы IBM / Дж. Радер, К. Миллсап : пер. с англ. – М. : Радио и связь, 1991. – 30 л.: ил.

4. Юркин, А. Г. Задачник по программированию / А. Г. Юркин. - СПб. : Питер, 2002. – 192 с.

5. Абрамов, С. А. Задачи по программированию / С. А. Абрамов, Г. Г. Гнездилова, Е. Н. Капустина, М. И. Селюн. – М. : Наука, 1988. – 224 с.

6. Абрамов, С. А. Начала информатики / С. А. Абрамов, Е. В. Зима. – М. : Наука, 1989. – 256 с.

7. Зуев, Е. А. Язык программирования Turbo Pascal 6.0, 7.0 / Е. А. Зуев. – М. : Веста, Радио и связь, 1993. – 384 с. : ил.

8. Довгаль, С. И. Персональные ЭВМ: ТурбоПаскаль V 7.0. Объектное программирование. Локальные сети : учебное пособие / С. И. Довгаль, Б. Ю. Литвинов, А. И. Сбитнев.

9. Епанешников, А. Программирование в среде Turbo Pascal 7.0 / А. Епанешников, В. Епанешников. – М. : «ДИАЛОГ-МИФИ», 1993. – 288 с.

10. Новичков, В. С. Паскаль : учеб. пособие для сред. спец. учеб. заведений / В. С. Новичков, Н. И. Парфилова, А. Н. Пылькин. – М. : Высш. шк., 1990. – 223 с. : ил. – (Алгоритмические языки в техникуме).

11. Хершель, Рудольф. Турбо Паскаль / Рудольф Хершель. – 2-е изд., перераб., – Вологда : МП «МИК», 1991. – 342 с. при участии МП ТПО «Квадрат», г. Москва.

12. Эрбс, Х.-Э. Введение в программирование на языке Паскаль : пер. с нем. / Х.-Э. Эрбс, О. Штольц. – М. : Мир, 1989. – 299 с., ил.

13. Григас, Г. Начала программирования : кн. для учащихся : пер. с лит. / Г. Григас ; под. ред. Ю. А. Первина. – М. : Просвещение, 1987. – 112 с. : ил.

14. Могилев, А. В. Информатика : учеб. пособие для студ. пед. вузов / А. В Могилев, Н. И. Пак, Е. К. Хеннер ; под ред. Е. К. Хеннера. – М., 1999. – 816 с.

15. Рубенкинг, Н. Турбо Паскаль для Windows : в 2 т. / Н. Рубенкинг ; пер. с англ. – Т. 1. – М. : Мир, 1993. – 536 с., ил.

16. Турбо Паскаль 7.0. – К. : Торгово-издательское бюро BHV, 1996. – 448 с.

17. Попов, В. Б. Turbo Pascal для школьников : учебное пособие / В. Б. Попов. – 3-е доп. изд. –М. : Финансы и статистика, 1999. – 528 с.

18. Немюгин, С. А. Turbo Pascal : практикум / С. А. Немнюгин. – СПб. : Питер, 2001. – 256 с.

19. Немюгин, С. А. Turbo Pascal : учебник / С. А. Немнюгин. – СПб. : Питер, 2001. – 496 с.

20. Культин, Н. Б. Программирование в Turbo Pascal 7.0 и Delphi / Н. Б. Культин. – 2-е изд., перераб. и доп. – СПб. : БХВ–Петербург, 2001. – 416 с.