ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ЛБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ»

Кафедра прикладной математики
С.Ж. Кишенский, О.В. Невельская

ИНФОРМАТИКА

часть II

ПОСОБИЕ
по выполнению лабораторных работ

для студентов
специальности 190701
дневного обучения

Москва – 2010

Рецензент Кучинская Н.В.

Кишенский С.Ж., Невельская О.В.

Пособие к лабораторным работам по дисциплине “ИНФОРМАТИКА”,часть II – М.:МГТУ ГА, 2010.- 107с.

Данное пособие издается в соответствии с учебным планом для студентов первого курса специальностей дневного обучения.

Рассмотрено и одобрено на заседании кафедры №3 от 17.11.2009г. и

методического совета по спец.190701 № 2 от 08.12. 2009г.

Содержание

Введение

Данные методические указания предназначены для студентов 1-го курса дневной формы обучения специальности 190700 (653400) и содержат описания 7 лабораторных работ. Методические указания могут использоваться в качестве учебного материала по аналогичным дисциплинам других специальностей.

В данном пособии рассматриваются вопросы разработки программ в среде ***Turbo Pascal***. Уделено внимание основным понятиям, операторам ввода и вывода данных, составлению программ, реализующих ветвление, циклические процессы, работе с массивами, процедурами, файлами и записями. Ко всем изучаемым темам прилагаются контрольные вопросы и задания для самостоятельной работы. Предполагается выполнение предложенных лабораторных работ во втором семестре после сдачи студентами первой части лабораторного практикума и приобретения ими навыков работы на компьютерах IBM PC.

Для успешного выполнения лабораторных работ данного сборника студенты должны уметь работать с операционной системой Windows, приобрести навыки работы в текстовых редакторах.

Выбор в качестве языка программирования алгоритмического языка Turbo Pascal объясняется тем, что он является развитым структурным и процедурным языком программирования.

Целью выполнения лабораторных работ сборника является приобретение практических навыков программирования различных типов алгоритмов, отладки программ на ПЭВМ и освоение системы программирования.

Интегрированная среда рассматриваемой системы имеет мощный экранный текстовый редактор, управляющую среду с многооконными меню, подсистему помощи, отладчик и встроенный компилятор.

Предполагается, что студенты приходят на занятия, предварительно изучив методические указания и подготовив программы индивидуальных заданий.

По каждой лабораторной работе подготавливается отчет, и работа защищается студентом.

Отчет должен содержать:

- конспект, в котором в краткой форме отражаются все разделы лабораторных работ;
- схемы алгоритмов решения заданий;
- программы, реализующие алгоритмы;
- выводы по работе.

Работа в интегрированной среде Turbo Pascal

Система программирования ***Turbo Pascal*** - интегрированная система, включающая текстовый редактор, управляющую среду с многооконными меню, подсистему помощи ***Help***, отладчик и встроенный компилятор.

После входа в Turbo Pascal на экране появляется окно редактора, в центре которого на сером фоне находится приглашение в систему.

Первая строка содержит все команды главного меню. В последней строке экрана приведены основные доступные в каждый текущий момент функциональные клавиши с указанием их назначения. Рабочее поле (окно редактирования) предназначено для вывода на экран и редактирования программы.

Окно редактирования имеет по периметру рамку. На рамке окна указывается:

- сверху слева закрывающая кнопка;
- сверху в середине путь и имя файла;
- внизу слева указывается местоположение курсора в редактируемой программе (первая цифра – номер строки, вторая – номер колонки текста).

Закрытие окна осуществляется щелчком левой кнопки мыши по закрывающей кнопке. Переход между программами, расположенными в различных окнах, осуществляется левым щелчком мыши выбором команды **WINDOW** и команды **Next**. Синоним [F6].

Строка меню Turbo Pascal

Строка меню **Turbo Pascal** (***TP***) активизируется нажатием функциональной клавиши [***F10***] или левым щелчком мыши. Строка меню содержит имена следующих меню:

File (файл) - позволяет выполнять все основные операции с файлами (создавать новые, загружать имеющиеся, сохранять созданные и отредактированные файлы, выводить на печатающее устройство содержимое этих файлов);

Edit (редактирование) - позволяет выполнять все основные операции редактирования текста (копировать, вставлять, удалять фрагменты текста, а также восстанавливать первоначальный вариант редактируемого текста);

Search (поиск/замена) – позволяет осуществлять поиск фрагментов текста и при необходимости производить замену найденного фрагмента новым;

Run (выполнение) - позволяет запускать программу, находящуюся в рабочей зоне, а также при необходимости пошагово выполнять данную программу или её часть;

Compile (компилирование) - позволяет осуществить компиляцию программы, которая находится в рабочей зоне;

Debug (отладка) - содержит команды, облегчающие процесс поиска ошибок в программе (**Breakpoints –** точки остановки, окно отладки **Watch,**

окно используемых программ, окно регистров, окно выходных результатов и некоторые другие);

Tools (сервис) - позволяет выполнять некоторые программы, не выходя из TP;

Options (параметры) - позволяет установить необходимые для работы параметры компилятора и TP;

Window (окно) - позволяет выполнить все основные операции с окнами (открывать, закрывать, перемещать, изменять размер);

Help (справка) - позволяет получить имеющуюся в системе справочную информацию.

Необходимое подчиненное меню активизируется (открывается) при помощи комбинации клавиш [***Alt*** + клавиша первой буквы имени подчиненного меню], а также путем последовательной активизации клавиш [***F10***] и первой буквы имени подчиненного меню. Выйти из подчиненного меню можно, нажав клавишу [***ESC***].

Диалоговые окна имеют несколько полей, ограниченных прямоугольной рамкой. Переход от одного поля к другому осуществляется щелчком мыши или нажатием клавиши ***TAB***. Курсор, находящийся в окне указывает на активное поле, в котором можно осуществлять необходимые записи, (например: записать путь к файлу). Выход из окна осуществляется нажатием клавиши ***ESC.***

Текстовый редактор системы Turbo Pascal

Окно редактирования предназначено для записи и редактирования программ с использованием встроенного текстового редактора системы ***Turbo Pascal***.

При загрузке системы вышеуказанным способом, автоматически устанавливается режим редактирования. После входа в режим редактирования курсор установлен в левом верхнем углу чистого поля экрана и показывает место, с которого можно набирать программу. Текст программы вводится с клавиатуры.

Далее перечислены наиболее часто употребляемые команды текстового редактора, выполняемые с помощью клавиш управления курсором:

[***Home***]	курсор переводится на начало строки;
[***End***]	курсор переводится на конец строки;
[***Ctrl+Home***]	курсор переводится на первую строку экрана;
[***Ctrl+End***]	курсор переводится на последнюю строку экрана;
[***PgUp***]	продвижение по файлу на одну страницу назад;
[***PgDn***]	продвижение по файлу на одну страницу вперед;
[***Ctrl+PgUp***]	курсор переводится в начало файла;
[***Ctrl+PgDn***]	курсор переводится в конец файла;
[***Ctrl+W***]	экран сдвигается «вверх» по тексту (при этом курсор неподвижен);

[***Ctrl+Z***]	экран сдвигается «вниз» по тексту (при этом курсор неподвижен);
[***Ctrl+Q***]+[***B***]	курсор переводится в начало блока;
[***Ctrl+Q***]+[***K]***	курсор переводится в конец блока;
[***Ctrl+Q***]+[***P***]	курсор перемещается на исходную позицию после поиска;
[***Ctrl+P***]	ввод специального символа;
[***Del***]	удаление символа, указываемого курсором;
[***Ins***]	переключение между режимами вставки и замены;
[***Backspace***]	удаление символа слева от курсора;
[***Ctrl+T***]	удаление слова справа от курсора;
[***Ctrl+Q***]+[***Y***]	удаление части строки от курсора до конца строки;
[***Ctrl+Y***]	удаление строки, указываемой курсором;
[***Ctrl+N***]	вставка строки.

Нормальный режим работы редактора - режим **вставки,** текстовый курсор имеет вид подстрочной черточки. В режиме вставки пропущенный символ вставляется перед тем символом, под которым установлен курсор.

Выполнение программы

После загрузки системы программирования необходимо:

- ввести текст программы;
- отладить программу;
- выполнить и получить результат.

После ввода текста программы следует:

а) войти в основное меню;

б) установить курсор на пункт **Run**;

в) в открывшемся подменю выбрать пункт **Run** и нажать **ENTER**.

Пункты а-в) можно заменить нажатием "горячих клавиш" - [**Ctrl+F9**].

После запуска программы на выполнение возможны две ситуации:

- в программе транслятор системы обнаружил ошибки;
- в программе ошибок не обнаружено.

Если в программе имеются ошибки, то сообщение о первой высвечивается в открывающемся на экране окне, а место ошибки отмечается в тексте программы курсором. Для продолжения работы нужно нажать клавишу **Esc**, исправить ошибку и снова запустить программу на выполнение.

Если больше ошибок не обнаружено, то программа выполняется и на экране появляется результат.

Создание и Сохранение программы

Создание новой программы выполняется через пункты **меню File, New**.

Загрузка имеющейся на диске программы производится через пункты меню **File**, **Open** ([***F3***]), затем в появившемся окне (**Files**) необходимо выбрать папку, где находится нужный файл, затем щелкнуть по кнопке **Open**.

Выбор диска можно произвести через пункты меню **File**, **Change Dir** и в появившемся окне **Change Directory** щелкнуть двойным левым щелчком по надписи **Driver** и выбрать соответствующий диск, нужную папку, а затем щелкнуть левым щелчком мыши по кнопке [***Ok*]**.

При выполнении программы, набранной непосредственно в редакторе, вся информация хранится в оперативной памяти ЭВМ, которую необходимо сохранить на диске.

Сохранение файла производят через пункты меню **File**, **Save**, [***F2*]**, а в появившемся окне задают имя файла, выбрают соответствующую папку, где необходимо сохранить файл, а затем щелкуют левым щелчком мыши по кнопке [***Ok*]**.

Сохраняйте программу по частям в процессе ее набора!

Компилирование имеющейся в окне программы производят через команду **Compile**, затем выбрают пункт меню **Compile** и нажимают клавишу [***Enter***]. [***Alt+F9***].

Завершить работу с ***TP*** можно с помощью комбинации клавиш [***Alt + X***] или команды **Quit** меню **File** (кратко – [***Alt + F],*** [***Q***]).

Если возникла необходимость временно выйти из ***TP,*** например, для ввода команд в ответ на подсказку ***MS-DOS***, вызовите команду ***File/DOS Shell.*** При этом ***TP*** останется в памяти, но управление будет передано ***DOS***. После выхода из TP Вы можете ввести команды ***DOS*** или запустить другие программы. Когда Вы будете готовы вновь вернуться в ***TP***, наберите в командной строке команду ***EXIT*** и нажмите клавишу [***Enter***]. При этом ***TP*** появится в том же состоянии, в котором была, когда Вы выходили из нее.

Лабораторная работа № 6. *Программирование линейных алгоритмов*.

Цель работы:

1. Приобретение навыков в составлении простейших программ на алгоритмическом языке Turbo Pascal.
2. Изучение среды программирования Turbo Pascal.
3. Приобретение навыков работы в редакторе Turbo Pascal.

Решение задачи с помощью ЭВМ состоит из следующих этапов:

- формулирования условия задачи;
- выбора метода ее решения;
- разработки алгоритма;
- составления программы на алгоритмическом языке;
- отладки программы.

В предлагаемых заданиях к лабораторным работам условия задач уже представлены в математической формулировке с указанием численного метода решения и необходимость в выполнении первых двух этапов отпадает, а также предложены задания для самостоятельной работы, где студенту потребуется

применить полученные знания и навыки.

Программа на языке Turbo Pascal - это последовательность строк, описывающих алгоритм решения задачи. Строка может содержать один или несколько операторов, разделенных точкой с запятой ";".

Структура программы Turbo Pascal

Program [имя программы]
Uses [имя модуля]
Label [описание меток]
Const [описание констант]
Type [описание типов]
Var [описание переменных]
Procedure [имя процедур]
Function [имя функций]
Exports [описание экспортируемых имен]
Begin
[тело программы]
end.

Модуль ***uses***

Uses system – загружается автоматически, включает в себя: файловый ввод/вывод, обработка строк, операции с плавающей точкой.

Uses graph – подключает специальные программы для работы в графическом режиме.

Uses crt – разрешает использование всех возможностей клавиатуры и дисплея (включая: управление экраном, цветом окна, звуковые сигналы).

Uses Dos – поддерживает функции MS DOS (установка тек. Времени, поиск по каталогам файлов).

Uses Printer – позволяет организовать доступ к устройству печати.

Label - (метка) - последовательность цифр 0 – 9999 или символов.

Пример: label 10, err;
Begin 10: оператор1; err: оператор2; end.

Const

Константы:

- ***литеры*** – числа (целые, вещественные), символы и строки символов.
- ***именованные константы*** – задание фиксированного значения при определении константы в начале программы с присвоением ему имени. Const <имя константы> = <значение константы>;

Пример: const dr := 9.81; max := 1000; hd := ‘табл.5’; pr := ‘ ‘;

Константы, которые можно использовать без предварительного описания: ***False*** ↔ ложь, ***True*** ↔истина, ***maxint*** ↔32767, ***pi***↔3.14159265358979, ***maxlongint***↔2147483647

• ***Типизированные константы.*** Данные константы могут изменять свое значение, поэтому их используют для начального присвоения значений.

Const *<имя константы>* ***: <****тип константы****> = <****значение****>;***

Пример:

const num : integer := 2007; ing : real := 19.90; zag : string [7] :='массив';

Оператор представляет собой строго формализованное указание на выполнение конкретного действия.

Строка может начинаться с метки. Метка может быть цифровой или буквенно-цифровой. Буквенно-цифровые метки могут иметь от 1 до 40 символов и начинаться с буквы, а завершаться двоеточием ":". Метка не определяет порядок выполнения строк программы, а служит для ссылки на нее. Программные строки выполняются в порядке их записи. Длина программной строки не должна превышать 256 символов.

Современные алгоритмические языки используют наборы различных типов данных.

Программа, написанная на языке Turbo Pascal, обрабатывает числовые и символьные данные. Данные представляются в программе в виде констант и переменных. Тип данных определяет возможные значения констант и переменных, форму представления в ЭВМ, объем занимаемой памяти, операции, которые могут выполняться над данными этого типа.

Числа. В языке Turbo Pascal пользуются двумя типами чисел: вещественными и целыми.

Под целое число отводится 1 байт или 2 байта памяти, и оно хранится форме с фиксированной точкой. Запись целого числа представляет собой последовательность цифр со знаком или без него (например:120, -13, 5487, -7821, +3841).

Вещественные числа хранятся в ячейке памяти длиной 4 байта в форме с плавающей точкой. Возможны две формы " внешней " записи вещественных чисел в программах:

- с фиксированной точкой (например, - 3.7);
- с плавающей точкой (например: -00.45E2, 0.78E-3, здесь буква " E означает основание " 10 " и разделяет мантиссу и порядок).

Числовое или символьное значение можно сохранить в переменной или в константе.

Переменная - величина, которая может меняться при выполнении программы. Переменная всегда имеет имя, которое содержит не более 40 буквенно-цифровых символов и начинается с латинской буквы. Переменные описываются явно с помощью операторов описания типа (табл.1).

Таблица 1

Таблица диапазона числовых данных

тип	диапазон
целый	-32768 ÷ +32767
целочисленный короткий	-128 ÷ +127
целочисленный длинный	-2147483648 ÷ +2147483647
веществ. обычной точности	-3.402823E+38 ÷ -1.40129E-45 +1.40129E-45 ÷ +3.402823E+38
веществ. двойной точности	-1.79769E+308 ÷ -4.94965E-324 +-4.94965E-324 ÷ +1.79769E+308

Программирование линейных алгоритмов.

Алгоритм - это последовательность действий, однозначно определяющих процесс преобразования исходных и промежуточных данных в результат решения задачи. Форма представления алгоритма может быть как текстовой, так и графической - в виде схемы. Решение всего многообразия задач может быть сведено к трем типам алгоритмов: линейному, разветвляющемуся и циклическому. Чаще встречается комбинация этих типов.

Линейный алгоритм - алгоритм, в котором к результату решения задачи приводит последовательное выполнение действий.

Алгоритм решения такой задачи в словесной форме состоит из следующих пунктов: начало программы; ввод исходных данных; вычисления; вывод результатов; окончание программы.

Программы, реализующие линейный алгоритм, как правило, очень просты и для их реализации используются операторы ввода данных, выполнения действий (вычислений) и вывода результатов.

Ввод данных

Ввод данных в программах, написанных на языке ***Turbo Pascal,*** можно осуществить с помощью оператора ***Read, Readln***:

Read (a, b, c), Readln (a, b, c) - операторы ввода данных в диалоговом режиме, где *a, b, c* **- *список имен переменных***

Встретив этот оператор, машина останавливает выполнение программы и ждет ввода данных. Значения переменных нужно ввести, нажав клавишу **Enter**. Количество, последовательность и тип вводимых данных должны соответствовать именам переменных оператора "**Read**", "**Readln**".

На экране можно вывести подсказку - текст, заключенный в одинарные кавычки в операторе "**Write**", "**Writeln**" например: **Write (‘** введите

значения a,b,c:’);. При выполнении такого оператора “**Write**”, “**Writeln**” на экране появится подсказка ***введите значения a,b,c:***, после чего нужно ввести значения переменных и нажать **Enter**.

Задать исходные данные можно также операторами присваивания, например: **a := 5 ; b := -8 ; c := 12.5;**

Выполнение вычислений

Для вычисления арифметических выражений используется оператор **присваивания “ := "**, частный случай которого применяется и для ввода данных.

Общий вид оператора: **Q := P ,** где **Q** - имя переменной, **P** - арифметическое выражение.

Арифметические выражения соответствуют общепринятым алгебраическим выражениям, в них могут входить числа, переменные, функции, соединенные знаками арифметических операций и круглыми скобками.

В языке Turbo Pascal имеются следующие арифметические операции:

*, / - умножение, деление;
MOD - определение остатка от деления;
DIV - целочисленное деление;
+, - - сложение, вычитание.

Операции перечислены в порядке убывания приоритета их выполнения. Действия внутри круглых скобок выполняются первыми.

Примеры записи некоторых арифметических операций и их результаты приведены в таблице 2.

Таблица 2

Результаты арифметических операций

ВЫРАЖЕНИЕ	РЕЗУЛЬТАТ	ПРИМЕЧАНИЕ
7 / 2	3.5	деление
7 DIV 2 8 DIV 4	3 2	операнды округляются, тип результата INTEGER
7 MOD 2 8.3 MOD 3.3 8 MOD 4	1 2 0	операнды округляются, тип результата INTEGER

Математические функции

При записи функций на языке Turbo Pascal *аргумент функции заключается в круглые скобки*. В качестве аргумента математических функций может быть число, переменная или арифметическое выражение. Следует отметить, что в тригонометрических функциях аргумент должен быть задан в радианах.

Наиболее распространенные функции языка **Turbo Pascal**:

Abs(X) - вычисляет модуль аргумента, что соответствует математической записи $|x|$;

Exp(X) - экспонента, соответствует математической записи e^x;

Ln(X) - вычисляет натуральный логарифм аргумента;

Sqr (x) - вычисляет возведение аргумента в квадратную степень;

Sqrt(X) - вычисляет корень квадратный из аргумента;

ArcTan(X) - вычисляет арктангенс аргумента;

Cos(X) - вычисляет косинус аргумента;

Sin(X) - вычисляет синус аргумента;

Tan(X) - вычисляет тангенс аргумента;

Sgn(X) - определяет знак аргумента. Если аргумент отрицательный, функция принимает значение (**-1**,) если положительное (**+1**). При нулевом аргументе функция принимает значение **0.**

Fix(X) - отбрасывает дробную часть значения аргумента;

Int(X) - округляет аргумент в сторону уменьшения;

Cint(X) - округляет аргумент по математическим правилам.

Примеры записи функций округления и их результаты в таблице 3

Таблица 3

Результаты функций округления

Выражение	Результат	Выражение	Результат	Выражение	Результат
FIX(5.7)	5	INT(5.7)	5	CINT(5.7)	6
FIX(5.1)	5	INT(5.1)	5	CINT(5.1)	5
FIX(-5.7)	-5	INT(-5.7)	-6	CINT(-5.7)	-6
FIX(-5.1)	-5	INT(-5.1)	-6	CINT(-5.1)	-5

Random(X) - выдает случайное число обычной точности в интервале 0÷1. Аргумент может быть опущен. Рекомендуется в начале программы запустить генератор случайных чисел оператором **Randomize**.

Для более подробного ознакомления с набором встроенных функций и их синтаксисом необходимо обратиться к документации по описанию языка программирования.

Вывод данных и результатов

Write (x, y, z); Writeln (x, y, z); - оператор вывода значений переменных **x, y, z.**

В качестве элементов вывода могут быть имена переменных, арифметические выражения, а также текст, заключенный в одинарные кавычки.

При выполнении оператора на экран выводятся значения переменных, арифметических выражений, текст. Список может отсутствовать и в этом

случае на экране пропускается строка.

Разделителем элементов вывода является запятая. Если вы используете оператор **Writeln (x, y, z);**, то очередной элемент выводится в начале следующей строки.

Задание 1.*Составить программу вычисления и вывода на экран радиусов описанной и вписанной окружностей R1 и R2 правильного многоугольника, а также площади правильного многоугольника. Количество сторон многоугольника - n и длину его стороны - a задать с экрана монитора.*

Для вычисления воспользуемся следующими формулами:

$R1 = \frac{a}{2\sin(3{,}14/n)}$ - радиус описанной окружности;

$R2 = \frac{a}{2tg(3{,}14/n)}$ - радиус вписанной окружности;

$S = \frac{n \cdot a \cdot R2}{2}$ - площадь правильного многоугольника.

Алгоритм решения задачи в словесной форме состоит из следующих пунктов: начало; ввод значений переменных **a** и **n**; вычисление функции R1, R2 и S; вывод значений функций R1, R2 и S; окончание программы.

```
Program zadacha_1;
  {Вычисления по формулам};
Var
R1, R2, S, a, n: real;
Begin
  writeln('введите значения a, n');
  readln(a, n);
  R1 :=a/(2*Sin(3.14/n));
  R2 = a/(2*Tan(3.14/n));
  S:= n*a*R2/2;
   writeln('Результат вычислений составил: ','R1=', R1, 'R2=',R2, 'S=', S);
End.
```

Пояснения к программе, которая реализует алгоритм**:**

- ***Program****;* начало программы;
- ***оператор {}*** позволяет вводить комментарии, пояснения к программе;
- ***Var*** оператор для описания переменных;
- ***Begin*** начало текста программ;
- **Writeln()** предназначен для вывода на экран текста, заключенного в апострофы, и значений переменных, а также выполнять вычисления искомых величин используется оператор присваивания «:=»;
- **Readln()** служит для ввода значений переменных по запросу;

- End. окончание программы.

Задание 2. *Составить программу вычисления по формуле и вывода на экран результата вычислений.* $z = 7.5e^{xy} + \sqrt[2]{\frac{5y}{7x}} + \log_9(xy)$

```
Program zadacha_2;
Var
rez, y, x, p: real;
Begin
writeln('введите значения y, x');
readln(y, x);
p := x*y;
rez:= 7.5*Exp(p)+Sqrt(5*y/(7*x))+Ln(p)/Ln(9);
 writeln('Результат вычислений =',rez);
End.
```

Пояснения к программе

В приведенной программе используются аналогичные операторы, что и в задании 1.

Следует обратить внимание на запись арифметического выражения:

-скобки определяют последовательность выполнения вычислений, количество открытых скобок равно количеству закрытых;

-для вычисления корня использована функция возведения в степень;

-для вычисления логарифма по основанию 9 используется математическая формула перехода.

Комментарий также можно записывать после апострофа.

Самостоятельная работа № 9

1. Изучить окно редактирования среды программирования **Turbo Pascal**.
2. Поочередно набрать тексты программ Вашего варианта.
3. Каждую программу сохранить в отдельном файле.
4. Программы отладить и получить результаты для различных исходных данных.
5. Результаты проанализировать.
6. Составить краткий конспект. Защитить работу.

Самостоятельная работа состоит из трех задач. Студент выбирает из списка заданий свой индивидуальный вариант и выполняет его.

Вариант №1

Задание 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$\sqrt{3x + xy - 6} * \ln(x + y) + tgy$

Задание 2.

Даны катеты прямоугольного треугольника *a* и *b*. Определить его гипотенузу, периметр и площадь.

Задание 3.

Определить число, получаемое выписыванием в обратном порядке цифр исходного трехзначного числа *f*.

Вариант № 2

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\lg\left|(y-\sqrt{x})*(x-\frac{y}{z+x^2/4})\right|$$

Задание № 2.

Определить периметр правильного *n*-угольника, если радиус вписанной окружности равен *r*.

Задание № 3.

Вычислить дробную часть среднего арифметического трех заданных положительных чисел *a*, *b*, *c*.

Вариант № 3

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\sqrt{(3x+1)}*(y+\cos x)^2-\ln(\frac{x+y}{x-y})$$

Задание № 2.

Найти площадь кольца, внешний и внутренний радиусы которого равны соответственно *R* и *r*.

Задание № 3.

Определить сумму цифр заданного четырехзначного числа *f*.

Вариант №4

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\frac{y+\sin x}{\lg xy}+(x-2y)^2$$

Задание № 2.

Даны три стороны произвольного треугольника *a*, *b*, *c*. Определить его площадь по формуле Герона.

Задание № 3.

Определить произведение первой и последней цифр заданного трехзначного числа *f*.

Вариант №5

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\lg(1+tg^2\frac{\sin xy}{e^x})$$

Задание № 2.

Вычислить расстояние между двумя точками на плоскости с координатами *(x1, y1)* и *(x2, y2)*.

Задание № 3.

Определить сумму квадратов цифр, входящих в заданное трехзначное число *f*.

Вариант №6

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$7{,}45e^{x+y}+6{,}2\sin\lg x$$

Задание № 2.

Вычислить периметр треугольника при заданных координатах его вершин *(x1,y1), (x2,y2), (x3,y3).*

Задание № 3.

В исходном трехзначном числе *f* удвоить число десятков.

Вариант №7

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$4Lnx\ +\log_3 7x^2-\frac{\sin(x+y)}{\cos(x-y)}$$

Задание № 2.

Вычислить площадь треугольника при заданных координатах его вершин *(x1,y1), (x2,y2), (x3,y3).*

Задание № 3.

Дано действительное число *x*. Определить сумму двух старших цифр в дробной части этого числа.

Вариант №8

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$2e^x+arctg\frac{1}{x}*\log_3 2y$$

Задание № 2.

Определить разность площадей круга и вписанного в него равностороннего треугольника.

Задание № 3.

Дано целое число *n<64*. Вывести его в двоичной системе счисления.

Вариант № 9

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\frac{x * arctgz + e^{-(x+y)}}{\log_3 2x + \sin(\lg x)}$$

Задание № 2.

Определить разность площадей квадрата и вписанного в него круга.

Задание № 3.

Дано действительное число *x* с трехразрядной дробной частью. Определить сумму двух младших цифр в дробной части этого числа.

Вариант № 10

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\log_3 y + \log_4 x^2 + \log_5 (xy)^2 + tgx$$

Задание № 2.

Идет *k*-я секунда суток. Определить, сколько полных часов и минут прошло с начала суток.

Задание № 3.

Определить произведение цифр заданного трехзначного числа *f*.

Вариант № 11

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$tg^2 x + 4\sqrt{y} + \log_3 7x^2$$

Задание № 2.

Вычислить расстояние между двумя точками в пространстве с координатами *(x1, y1, z1)* и *(x2, y2, z2)*.

Задание № 3.

Определить произведение первой и последней цифр заданного трехзначного числа *f*.

Вариант № 12

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры): $\lg(1 + tg^2 \frac{\sin xy}{e^x})$

Задание № 2.

Дана длина ребра куба *a*. Найти объем куба и площадь его боковой поверхности.

Задание № 3.

Определить произведение двух первых цифр заданного четырехзначного числа *f*.

Вариант №13

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\cos^2 arctg\frac{1}{z} * e^{x-y}$$

Задание № 2.

Определить площадь и высоту равностороннего треугольника, периметр которого равен *f*.

Задание № 3.

Определить произведение двух последних цифр заданного трехзначного числа *f*.

Вариант № 14

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$y\sin x \ + e^{2x} - 5x\ln 5 \ + \sqrt{|x|}$$

Задание № 2.

Определить разность площадей круга и вписанного в него квадрата.

Задание № 3.

Определить произведение второй и третьей цифр заданного трехзначного числа *f*.

Вариант №15

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$x - \frac{\log_3 tgx^2}{1+\sin^2(x+y)} * e^x$$

Задание № 2.

Дана площадь куба. Найти ребро куба и его объем.

Задание № 3.

Определить сумму второй и третьей цифр заданного трехзначного числа *f*.

Вариант №16

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$\lg\left|(y-\sqrt{x}) * (x - \frac{e^y}{z + x^2 * 4})\right|$$

Задание № 2.

Дано число *x*, указывающее количество минут. Определить количество часов и минут.

Задание № 3.

Определить сумму первой и последней цифр заданного трехзначного числа *f*.

Вариант № 17

Задание № 1.

Вычислить выражения (значения буквенных переменных задавать с клавиатуры):

$$x+\frac{e^{y}+(xy)^{2}}{7x^{2}y+xy^{2}+\cos(3/4x)}-\sin x$$

Задание № 2.

Идет *k*-я секунда суток. Определить, сколько полных часов прошло с начала суток.

Задание № 3.

По заданному номеру года определить номер столетия, на которое приходится этот год.

Лабораторная Работа № 7. ***Программирование разветвляющих алгоритмов*.**

Цель работы:

1. Дальнейшее изучение приемов программирования на алгоритмическом языке **Turbo Pascal**.
2. Программирование условных алгоритмов.
3. Дальнейшее изучение среды программирования и приемов отладки программ.

***Программирование разветвляющих алгоритмов*.**

Алгоритм разветвляющейся структуры - алгоритм, в котором последовательность выполнения действий зависит от каких-либо условий.

В языке ***Turbo Pascal*** для ветвления используются следующие операторы:

1.Оператор безусловной передачи управления **GOTO N** , - где N метка строки. Этот оператор передает управление строке с меткой N.

2. Операторы условной передачи управления (приведены три типа):

а) ***IF*** < логическое выражение > ***THEN*** < оператор >**;**

- при выполнении оператора IF сначала определяется результат логического выражения: ИСТИНА (TRUE) или ЛОЖЬ (FALSE). Если ИСТИНА, то управление передается операторам, следующим за словом THEN, если - ЛОЖЬ, то оператору, записанному после оператора IF.

б) ***IF*** <логическое выражение>***THEN***< операторы >***ELSE***<оператор>**;**

- при выполнении оператора IF данной модификации сначала

определяется результат логического выражения. Если ИСТИНА, то управление передается операторам, следующим за словом THEN, если - ЛОЖЬ, то оператору записанному после ELSE.

в) блочный " **IF** " (записывается в нескольких строках):

IF < логическое выражение > **THEN** < оператор>
[ELSE IF < логическое выражение > **THEN** < оператор>
ELSE < оператор> **]*;**

При выполнении блочного IF, сначала определяется результат первого логического выражения. Если результат - ИСТИНА, то управление передается операторам, следующим за первым словом THEN, а затем к строке следующей за разделителем «;». Если - ЛОЖЬ, то определяется результат следующего логического выражения, и в случае ИСТИНЫ управление передается операторам, записанным за следующим THEN, а потом к строке идущей за «;» и т.д. Если же ни одно из условий оператора не выполняется, то выполняются операторы, записанные после слова ELSE, а потом управление передается строке, следующей за «;».

В командах разрешено использовать только *один оператор*, но если вам необходимо в команде написать более одного оператора, тогда применяют конструкцию **begin … end;** , которая позволяет *записать любое количество операторов*.

Например: **If a>=d Then begin b :=4; c := 5;**
d := a+b-c; End;

Логические выражения состоят из числовых или текстовых данных, знаков отношений и логических операций (табл. 4).

* * при записи структуры операторов в общем виде квадратные скобки означают необязательность элемента, заключенного в **“ [ ] ”**

Таблица 4

Логические операции

ЗНАКИ СРАВНЕНИЯ		ЛОГИЧЕСКИЕ ОПЕРАЦИИ	
Название знака	В программе	Название операции	В программе
Равно	=	Логическое умножение	**AND**
не равно	<>		
больше	>	Логическое сложение	**OR**
больше или равно	>=		
меньше	<	Отрицание	**NOT**
меньше или равно	<=		

Задание № 1. *Из двух случайных чисел x,y вывести наибольшее и если x>y также вывести разность этих чисел (рис.1).*

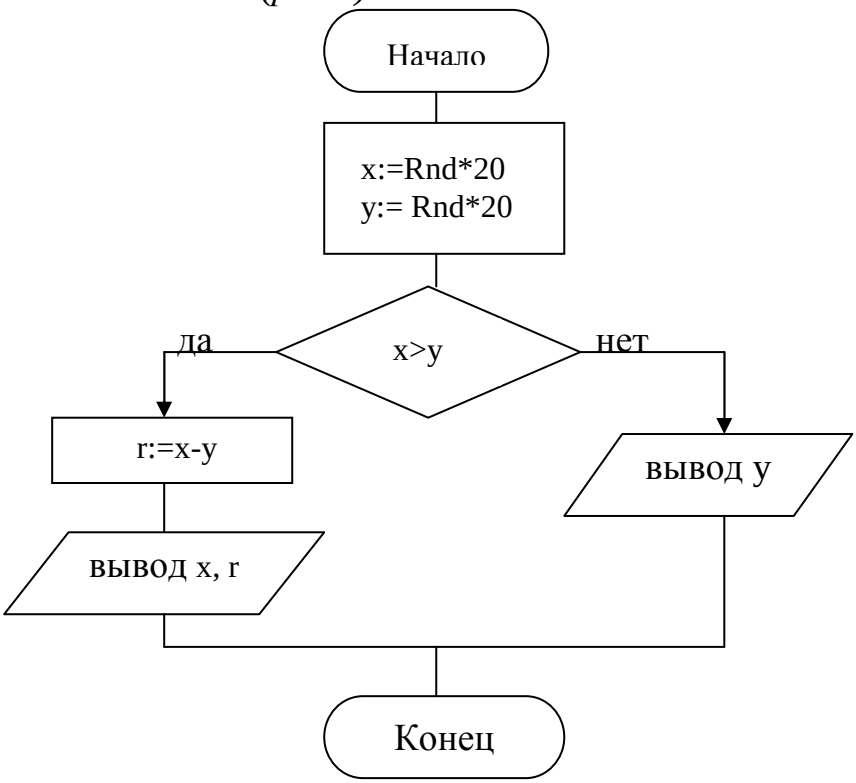


Рис. 1. Схема алгоритма к заданию №1

```
Program zadacha_1;
{Программирование разветвляющего алгоритма};
Var
rez, y, x, p: real;
Begin
Randomize;          {включен датчик случайных чисел};
x := Random(20);    {получение случайного числа x};
y := Random (20);   {получение случайного числа y};
If x > y Then begin
                    p := x – y;
```

```
                    Writeln (‘разность чисел равна’, p, ‘x=’, x);
                    End;
                Else Writeln (‘y=’, y);
End.
```

Пояснения к программе

В данной программе использован оператор **IF** - блочная форма. При выполнении условия **x>y** после **Then** в соответствии со схемой алгоритма записаны два оператора, разделенные «**;**»- оператор присваивания **r:=x-y** (разность чисел **x** и **y**) и вывода результатов. Если условие не выполняется, то выводится **y** после **Else**.

Задание №2. *Составить схему алгоритма(рис.2) и программу вычисления и печати функции F(x) для заданного значения x*:

$$F(x) = \begin{cases} \sin x, & \text{если } x \le a \\ \cos x, & \text{если } a < x < b \\ tgx, & \text{если } x \ge b \end{cases}$$

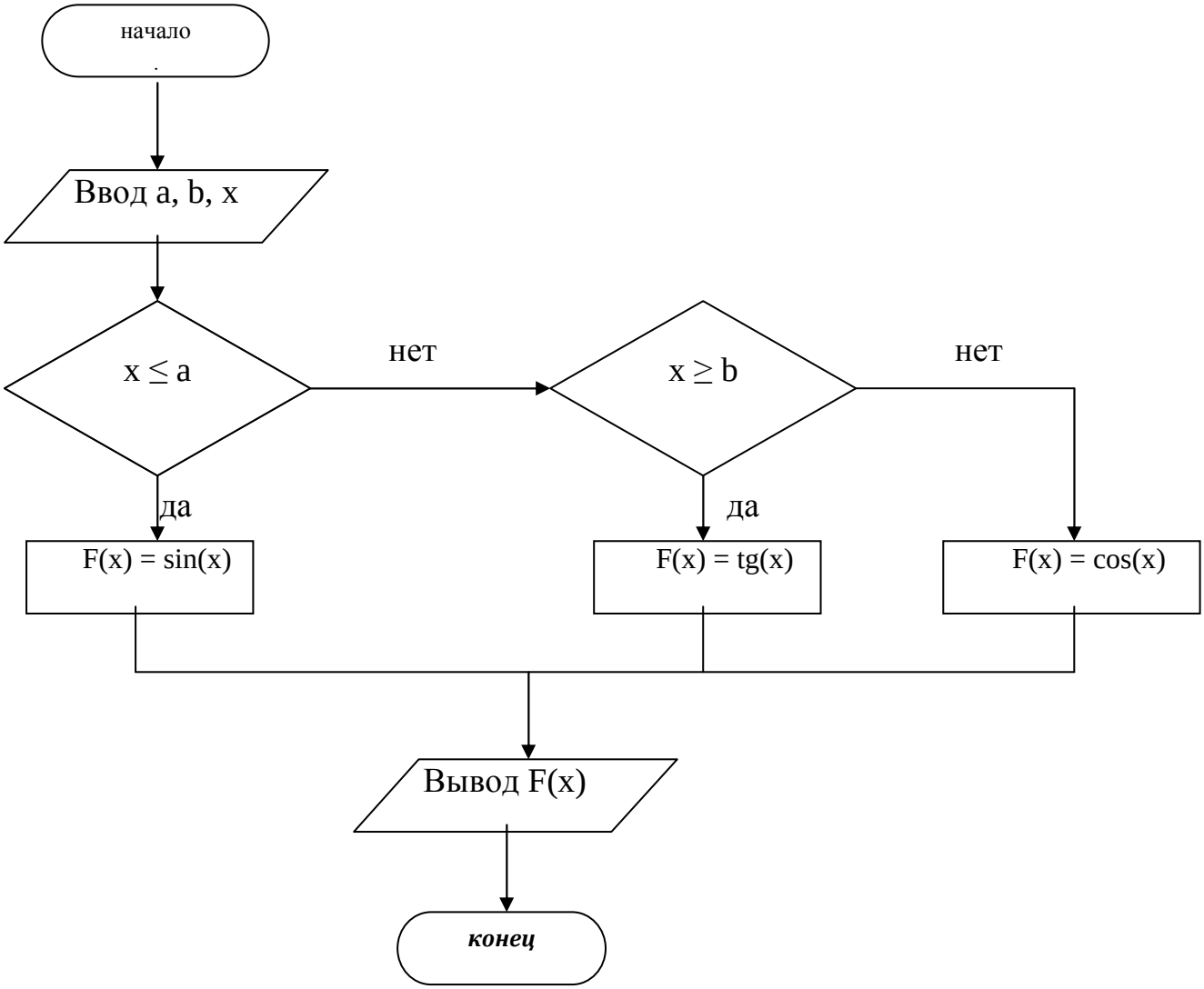


Рис. 2 Схема алгоритма к заданию №2

```
Program zadacha_2;
{Программирование разветвляющего алгоритма};
Var
f, a, x, b: real;
Begin
   Writeln ('Введите значения a, b, x');
   Readln (a, b, x);
   If x <= a Then    f := Sin(x);
                     Else if  x >= b Then f := Tg(x)
                              Else f := cos(x);
   Writeln ('x=', x, 'f=', f);
   End.
```

Пояснения к программе

В программе для организации ввода данных в диалоговом режиме используется только оператор " **Readln** " .

" **If** " - блочный обеспечивает ветвление. В зависимости от введенных значений переменных **a, b, x** процесс вычисления **f** пойдет в соответствии с алгоритмом (рисунок 5) по одной из ветвей.

Самостоятельная работа № 10

1. Набрать, отладить и выполнить программы, реализующие условные алгоритмы Вашего индивидуального задания.
2. Составить блок-схему.
3. Проанализировать работу операторов, пользуясь отладочными режимами.
4. Составить краткий конспект. Защитить работу.

Самостоятельная работа состоит из трех задач. Студент выбирает из списка заданий свой индивидуальный вариант и выполняет его.

Вариант № 1

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases}\max(x,y), \textit{если } x<0\\ 3y+1, \textit{если } x=0\\ y-\sin x, \textit{если } x>0\end{cases}$$

Задание № 2.

Даны действительные положительные числа *x, y, z.* Выяснить, существует ли треугольник с такими длинами сторон.

Задание № 3.

Определить, есть ли среди цифр заданного трехзначного числа *f* одинаковые.

Вариант № 2

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases}\min(x,y), \textit{если } x<=-1{,}75\\ xy, \textit{если } x<=0{,}28\\ 3x+y-6, \textit{если } x>=0{,}28\end{cases}$$

Задание № 2.

Для произвольных чисел *a, b, c* определить, имеет ли уравнение $ax^2+bx+c=0$ решение.

Задание № 3.

Определить, есть ли среди первых трех цифр дробной части заданного числа *f* цифра 0.

Вариант № 3

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} |xy|, \text{ если } x<y \\ 74x*(y-9), \text{ если } x=y \\ \max(x,23), \text{ если } x>y \end{cases}$$

Задание № 2.

Даны действительные числа *x*, *y*. Определить, принадлежит ли точка (*x*, *y*) кольцу с внутренним и внешним радиусами соответственно *R1* и *R2* и центром в начале координат.

Задание № 3.

Дано натуральное трехразрядное число *n*. Определить, содержит ли оно две одинаковые цифры.

Вариант № 4

Задание № 1

Вычислить *f(x, y).* Значения *x*, *y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} x^3-\sin^2 y, \text{ если } x<y \\ \min(x,0), \text{ если } x=y \\ x-|y|, \text{ если } x>y \end{cases}$$

Задание № 2

Дано натуральное число *n* (возраст).Обеспечить вывод этого числа с соответствующим словом:

«год», «года», «лет».

Задание № 3

Даны действительные числа *x*, *y*, *z*. Получить *max(x, y, z).*

Вариант № 5

Задание № 1.

Вычислить *f(x, y).* Значения *x*, *y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} \max(xy\ , y/7), \text{ если } x<y \\ x+\sin y, \text{ если } x=y \\ \frac{1}{x}+\frac{1}{y}, \text{ если } x>y \end{cases}$$

Задание № 2.

Даны действительные числа *x*, *y*. Определить, принадлежит ли точка (*x*, *y*) кругу с радиусом *R* и центром в начале координат.

Задание № 3.

Задано трехразрядное число *a* с двумя разрядам в дробной части. Определить, есть ли в этом числе цифра 8.

Вариант № 6

Задание № 1.

Вычислить *f(x, y).* Значения *x*, *y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} x-|y|, \text{ если } x>y \\ x+\sin(y), \text{ если } x=y \\ \min(x*y\ ,y+x), \text{ если } x<y \end{cases}$$

Задание № 2.

Задан номер года *n*. Определить, високосный ли он.

Задание № 3.

Задано трехразрядное число *a*. Определить, кратно ли оно 9.

Вариант № 7

Задание № 1.

Вычислить *f(x, y)*. Значения *x*, *y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} y\ -5x, \text{ если } 2x+3>0 \\ \max(x,y), \text{ если } 2x+3<0 \\ 3x+y^2+6, \text{ если } 2x+3=0 \end{cases}$$

Задание № 2.

Даны числа *a1*, *b1*, *c1*, *a2*, *b2*, *c2*. Определить координаты точки пересечения прямых $a1*x+b1*y=c1$ и $a2*x+b2*y=c2$, либо сообщить, что эти прямые совпадают или не пересекаются.

Задание № 3.

Определить, есть ли в заданном четырехразрядном числе ***a*** цифры 7 и 8.

Вариант № 8

Задание № 1.

Вычислить *f(x, y)*. Значения *x*, *y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} x(x*y\ -1), \text{ если } x>10 \\ \min(x,y), \text{ если } 0<x<=10 \\ 7+y, \text{ если } x<=0 \end{cases}$$

Задание № 2.

Даны действительные числа *a*, *b*, *c*. Выяснить, имеет ли уравнение $ax^2+bx+c=0$ корни. Если да, то найти их.

Задание № 3.

Дано натуральное трехзначное число ***a***. Выяснить, является ли это число «зеркальным» (например, 545, 131).

Вариант № 9

Задание № 1.

Вычислить *f(x, y)*. Значения *x*, *y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} \min(5x,6y), \text{ если } x>4 \\ 6xy+4, \text{ если } 2<x<=4 \\ x+\sin(y), \text{ если } x<=2 \end{cases}$$

Задание № 2.

Даны два круга с центрами в начале координат и радиусами *R1*, *R2 (R1>R2)*. Определить, принадлежит ли точка с координатами (*x*, *y)*

образованному кругами кольцу.

Задание № 3.

Задано трехразрядное число *a*. Определить, кратно ли оно 3.

Вариант № 10

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} xy, \text{ если } x>y \\ \min(xy, x+y), \text{ если } x<y \\ 74, \text{ если } x=y \end{cases}$$

Задание № 2.

Даны три точки на плоскости с координатами *(x1,y1), (x2,y2), (x3,y3).* Вывести номера точек с наибольшим расстоянием между ними.

Задание № 3.

Определить, есть ли в заданном четырехразрядном числе ***a*** цифры 3 и 4.

Вариант № 11

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} 4x-2(xy-1)\ -24, \ x<=0 \\ \max(x,12), \ x>5 \\ 5y+4, \ 0<x<=5 \end{cases}$$

Задание № 2.

Определить, есть ли среди младших трех цифр целой части заданного числа *s* цифра 5.

Задание № 3.

Даны числа **a, в, с.** Найти наибольшее среди них.

Вариант № 12

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} \min(x+y, xy), \text{ если } x>y \\ \dfrac{xy}{x+y}, \text{ если } x<y \\ 25, \text{ если } x=y \end{cases}$$

Задание № 2.

Даны действительные числа *x, y, z.* Вычислить значение *min(2(x+y+z),xyz ,(x+y)z)+1*.

Задание № 3.

Даны числа **a, в, с.** Найти наименьшее среди них.

Вариант № 13

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} yx\ -5x, \textit{ если } x>y \\ x+\sin(y), \textit{ если } x=y \\ \max(2x,y), \textit{ если } x<y \end{cases}$$

Задание № 2.

Дана точка с координатами (*x, y)*. Определить, какому квадранту плоскости она принадлежит.

Задание № 3.

Определить, равна ли сумма первых двух цифр четырехразрядного числа *f* сумме двух его последних цифр.

Вариант № 14

Задание № 1.

Вычислить *f(x, y)*. Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} \frac{1}{x}+\frac{1}{y}, \textit{ если } x-0{,}5>y \\ xy(x+y), \textit{ если } x<y+0{,}5 \\ \min(x,10), \textit{ если } x-0{,}5=y \end{cases}$$

Задание № 2.

Определить, есть ли в заданном четырехразрядном числе *f* цифра 3.

Задание № 3.

На поле *(a, b)* шахматной доски расположен ферзь. Определить, угрожает ли он полю *(c,d)*.

Вариант № 15

Задание № 1.

Вычислить *f(x, y)*. Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases} \left|\min(x,y+10)\right|, \textit{ если } x>y \\ 5-y, \textit{ если } x<y \\ \log_9(xy), \textit{ если } x=y \end{cases}$$

Задание № 2.

Даны действительные числа *x1, y1, x2, y2* (*x1>0, y1>0)*. Определить квадранты плоскости, в которых лежит отрезок, концами которого являются точки *(x1, y1)* и *(x2, y2)*.

Задание № 3.

Определить, делится ли заданное натуральное число *n* на 9.

Вариант № 16

Задание № 1.

Вычислить *f(x, y)*. Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases}\left(\frac{x}{3}\right)*y^2, \textit{если } x>y\\ \max(5x,2y), \textit{если } x<y\\ y^2-6x, \textit{если } x=y\end{cases}$$

Задание № 2.

Даны действительные числа *x1*, *y1*, *x2*, *y2* (*x1<0, y1<0).* Определить квадранты плоскости, в которых лежит отрезок, концами которого являются точки *(x1, y1)* и *(x2, y2).*

Задание № 3.

Определить, есть ли в заданном четырехразрядном числе *a* цифры 5 или 8.

Вариант № 17

Задание № 1.

Вычислить *f(x, y).* Значения *x, y* задаются с клавиатуры:

$$f(x,y)=\begin{cases}\max(x,y), \textit{если } x>10\\ |x-y|, \textit{если } x<=5\\ \frac{xy}{x+y}, \textit{если } 5<x<=10\end{cases}$$

Задание № 2.

Дан квадрат со стороной *b*, точка пересечения диагоналей которого расположена в начале координат. Определить, находится ли точка с координатами *(x, y)* внутри квадрата.

Задание № 3.

Определить, есть ли в заданном четырехразрядном числе *a* цифры 1 и 5.

Лабораторная Работа № 8. Программирование циклических алгоритмов.

Цель работы:

1. Изучение приемов программирования циклических алгоритмов.
2. Программирование циклических алгоритмов на языке **Turbo Pascal**.
3. Отладка циклических программ в среде программирования **Turbo Pascal.**

Циклические алгоритмы

Алгоритм называется циклическим, если все или отдельные его этапы в процессе решения задачи неоднократно повторяются.

Цикл обеспечивает повторное выполнение, или, иначе говоря, циклическую работу операторов. Оператор или группа операторов, повторяющаяся в цикле, называется "телом цикла".

Далее рассмотрим два типа циклических задач:

а) задачи, в которых вычисления многократно ведутся по одним и тем же формулам с различными значениями входящих в нее величин.

б) задачи, где значение некоторой величины вычисляется через значение этой же величины, полученное в предыдущем цикле (рекурсии). Примерами таких задач являются задачи вычисления сумм и произведений рядов, а также вычисление значений факториала.

Характерные моменты циклического алгоритма:

- *первоначальный вход в цикл выполняется через блок подготовки;*

- *цикл всегда характеризуется некоторой переменной, называемой параметром цикла. Начальное значение параметра задается перед циклом в блоке подготовки, а при каждом повторении цикла выполняются операторы тела цикла и параметр изменяется на единицу;*

- *число повторений цикла должно быть конечным, однако, не всегда число повторений известно или может быть вычислено заранее. Выход из цикла осуществляется при выполнении некоторых условий. Когда число повторений известно или может быть определено заранее, выход их цикла осуществляется при достижении параметром некоторой заранее заданной величины. Для такого рода задач используется оператор цикла "**FOR**". В общем виде оператор цикла записывается следующим образом:*

For I: = n TO k do <оператор1>;

или

For I := n TO k do begin

<оператор1>;

<оператор2>;

……

<оператор N>; End;

Здесь:

I - параметр цикла (переменная),

n - начальное значение параметра цикла (переменная или число),

k- конечное значение параметра цикла (переменная или число).

Этот оператор многократно выполняет <оператор>, находящийся после ключевого слова ***do*** для всех значений параметра **I** от **n** до **k** (при этом ***I =I+1, n< k***).

Внимание. Если параметр ***TO*** заменить на параметр ***DOWNTO***, то значение параметра ***I*** вычисляется по формуле ***I=I – 1 (n > k)***.

Структура самого оператора включает и подготовку цикла, и изменение параметра, и проверку условия выхода из цикла.

Проиллюстрируем использование оператора " ***For*** " в следующих заданиях.

Задание № 1. *Составить схему алгоритма (рис.3) и программу вычисления всех значений функции F(x) для всех значений аргумента x:*

F(x) = SIN(x)+COS(x), при -10 ≤ x ≤10, шаг изменения аргумента Δx=2

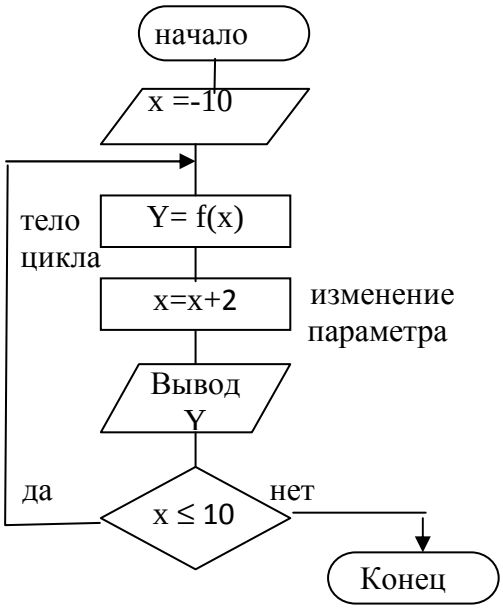


Рис. 3. Схема алгоритма к заданию №1

```
Program zadacha_1;
{Программирование
циклического алгоритма};
Var
y: real;
x: integer;
Begin
For x:=-10 to 10 do begin
     Y:=Sin(x)+Cos(x);
    Writeln ('x= ', x, ' y=',y);
    x:=x+1;    End;
End.
```

Пояснение к программе

В цикле ***For*** многократно вычисляются значения функции для всех значений аргумента. Значения аргумента и проверка условия выхода из цикла осуществляется самим оператором ***For***. В роли счетчика используется переменная ***x*** , которая автоматически увеличивается на ***1*** после каждого исполнения цикла ***For*** и дополнительно прибавляется ***1*** при срабатывании оператора ***x:=x+1;*** .

Задание № 2 .Составить схему алгоритма (рис.4) и вычислить сумму n слагаемых ряда:

$$S = 1 + \frac{1}{3} + \frac{1}{6} + \frac{1}{9} + ... + \frac{1}{3n} = 1 + \sum_{i=1}^{n} \frac{1}{3i}$$

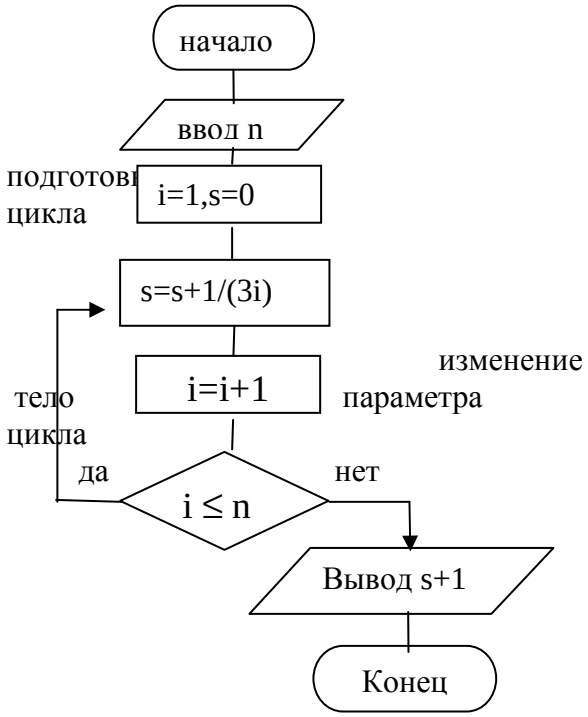


Рис. 4 Схема алгоритма к заданию №2

```
Program zadacha_2;
{Цикл с известным числом повторений};
Var
i , n : integer;
s : real;
begin
Write ('Введите кол-во слагаемых');
Read (n);
S:=1;
For i=1 to n do s=s+1/(3*i) ;
Writeln ('Сумма ряда:', s+1);
End.
```

Результат выполнения на экране:

***Сумма ряда*: < *число* >**

Задание № 3. Составить схему алгоритма (рис.5) и вычислить факториал n чисел: P= n!=1*2*3*….*n

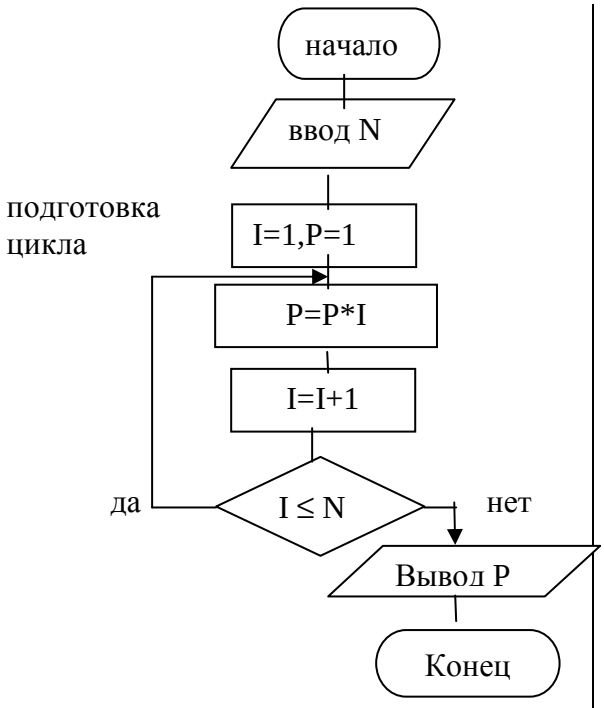


Рис. 5 Схема алгоритма к заданию №3

```
Program zadacha_3;
{ Цикл с известным числом повторений};
Var
i , n , p : integer;
Begin
Writeln ('Введите Число сомножителей');
Read (n);
p := 1;
For i=1 to n do p :=p*i ;
Write ('n факториал :', p);
End;
```

Результат выполнения на экране:

***n факториал* : < *число* >**

Пояснение к программам №2 и №3

С помощью циклов в этих программах последовательно вычисляется значение суммы и факториала (произведения). При этом каждое следующее значение вычисляется через предыдущее. Результатом многократных вычислений является одна величина - сумма ряда заданного числа слагаемых (***задание №2***) или произведение заданного числа сомножителей (***задание №3***).

В приведенных заданиях №1 , №2 и №3 число повторений легко определяется заранее. В решении задания №1 параметром цикла является переменная "***x***", а в заданиях №2 и №3 - переменная "***i***".

Когда условие задачи не позволяет определить заранее число повторений цикла, выход из цикла зачастую осуществляется по достижении заданной точности вычислений или по какому-либо другому условию, оговоренному в задании. В этих случаях удобно использовать следующие операторы цикла:

While **L do <*оператор*>;**	***Repeat*** **< *операторы* >;** ***Until L***

Здесь: L - логическое выражение.

Операторы, находящиеся между после ***do*** и после ***Repeat*** повторяются до тех пор, пока выражение, стоящее после ***While*** - истинно или до тех пор, пока выражение, стоящее после ***Until*** - ложно.

Задание № 4. *Составить схему алгоритма (рис.6) и вычислить сумму* ***s*** *ряда с заданной точностью* ***e*** *и количество слагаемых* ***n,*** необходимых для достижения данной точности: $$S = 1 + \frac{1}{3} + \frac{1}{6} + \frac{1}{9} + ... + = 1 + \sum_{i=1}^{\infty} \frac{1}{3i}$$

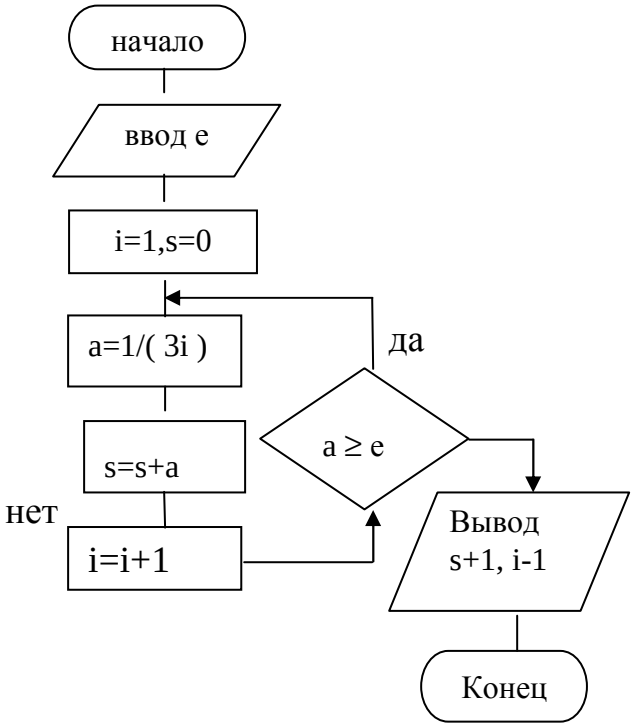


Рис. 6 Схема алгоритма к заданию №4

```
Program zadacha_4;
{Цикл с неизвестным числом повторений};
Var
e, i : integer;
s, a :real;
Begin
Write ('Введите точность e');
Read (e);
s := 0;  i:=1;
While a >= e do begin a := 1/( 3*i );
                      s := s+a ;
                      i :=i+1;
                     End;
Write ('Сумма ряда:', s+1,'n=', i-1);
End.
```

Результат выполнения на экране: ***Сумма ряда:<число> n=<число>***

Вычислить сумму ряда с точностью **e** - это значит производить вычисления до тех пор, пока очередной член последовательности не станет меньше или равен заданной точности.

В заданиях №2 и №4 требуется вычислить сумму ряда. В первом случае количество слагаемых определено условием задачи, а во втором - слагаемое последовательно, в цикле, прибавляется к сумме до тех пор, пока его абсолютная величина не станет меньше или равна некоторого маленького числа (заданной точности). Такая постановка задачи имеет смысл только для **сходящихся** рядов, т.е. слагаемое должно стремиться к нулю.

Далее приведено еще несколько заданий циклических программ.

Задание №5. *Составить схему алгоритма (рис.7) и найти корень уравнения* $3x^3 + 5x^2 - 6.9 = 0$ *методом деления отрезка пополам с точностью* ***e*** *на интервале* ***[a,b]***

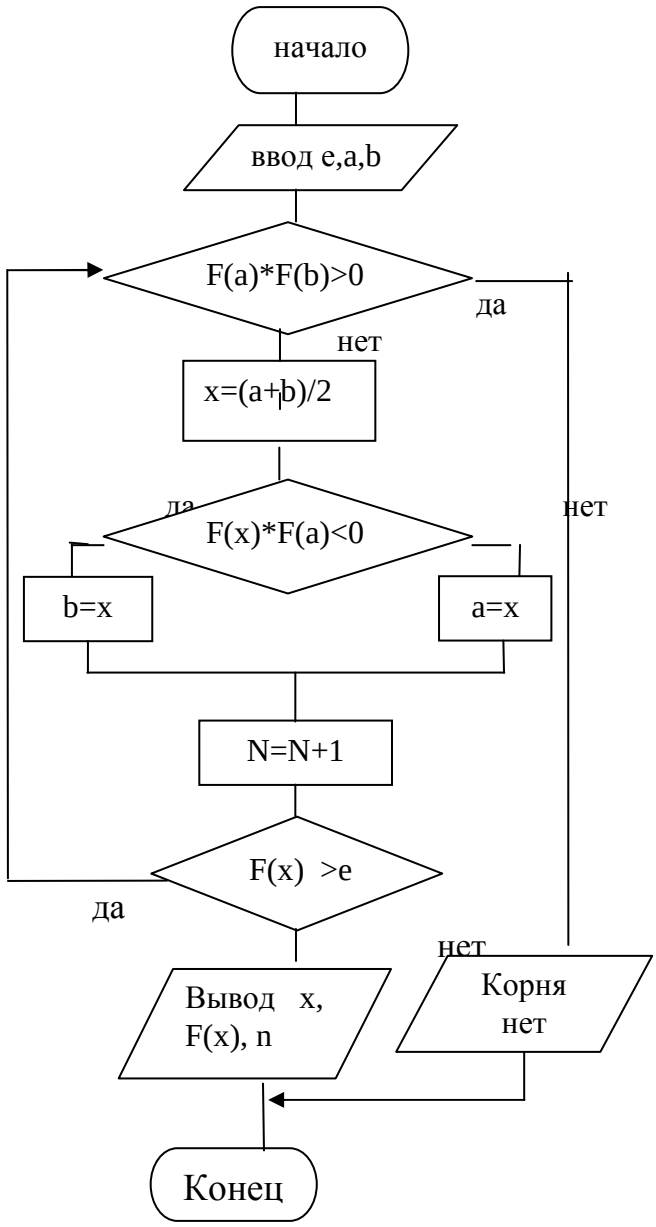


Рис. 7 Схема алгоритма к заданию №5

```
Program zadacha_5;
{ Цикл с неизвестным числом};
{Повторений};
Var
e, a, b, x, Fx, Fa, Fb: real;
n: integer;
Begin
Writeln ('Введите точность –e');
Readln (e);
Writeln ('Задайте интервал a,b');
Readln (a,b);
{Определим значения функции};
{на концах интервала};
Fa: = a * Sqr(a) * 3 + 5 * Sqr(a) - 6.9;
Fb : = a * Sqr(a) * 3 + 5 * Sqr(a) - 6.9;
{если знаки Fa и Fb одинаковы,};
{то корня на интервале нет};
If Fa * Fb > 0 Then  begin
                Writeln('корней нет');
                goto M1;    End;
Repeat
x:=(a+b)/2;
{Делим отрезок пополам};
{подставляем значение x};
  Fx:= x * Sqr(x) * 3 + 5 * Sqr(x) - 6.9;
{если корень на отрезке a – x, то};
{смещаем интервал влево, иначе - };
{вправо};
If  Fa * Fx < 0 then   b := x
                    Else   a := x;
   {считаем число приближений n};
        n := n + 1;
       Until Abs(Fx) > e
    Writeln( ' корень уравнения: ',x);
Writeln ('Fx=', Fx, 'n=',n);
M1: End.
```

Задание №6. *Найти максимум функции Y=SIN(X)*EXP(X+2.5)/(X-0.9) на интервале -5 < X < 3*

```
Program zadacha_6;
{максимум функции обозначим переменной  Ymax};
Var
Ymax, y: real;    x: integer;
```

```
Begin
{Зададим начальное значение переменной Ymax};
Ymax: = -1E+19 ;
For x = -5 to 3 do begin                {Сравним в цикле текущее};
y: = Sin(x) * Exp(x + 2.5) / (x -0.9);   {Значение функции y  с  Ymax};
 If y > Ymax then Ymax := y;       {и если ymax окажется меньше};
   End;                                {то его  значение  заменяется на текущее};
Writeln('YMAX= ', Ymax);     {выводим значение максимума функции};
 End.
```

Задание №7. *Определить является ли случайное число* ***x*** *простым?*

```
Program zadacha_7;
Var
x, x_in: real;
j, priz : integer;
Begin
{Задаем  целое число с помощью датчика случайных чисел};
Randomize ;
x_in:=Fix(Random(100));
x:=x_in/2;
priz:=0;
{ В цикле определяем делится ли введенное число на какое-либо};
{другое  кроме  1  и  самого  себя  без  остатка,  если  делится,  то  оно  не
простое};
For j := 2 TO x do begin
If x  Mod  j = 0 Then  priz:=1;
End;
If priz = 1 then Writeln ('простое число: ', x_in);
   Else Writeln (x_in, ' - это число не простое');
End.
```

Сложные циклы

Цикл называется сложным, если он содержит в себе другой, вложенный в него цикл. Количество вложенных друг в друга циклов (глубина вложений) может быть достаточно большим. Каждому циклу соответствует свой параметр. Типы циклов, из которых образован сложный, могут быть различными, это зависит от конкретной задачи. Первоначальный вход в любой цикл допустим только через блок подготовки соответствующего цикла. В общем виде схема алгоритма сложного цикла приведена на рисунке 8. Тело цикла включает в себя операторы соответствующего цикла. Причем для каждого значения параметра внешнего цикла параметр внутреннего цикла пробегает все свои значения.

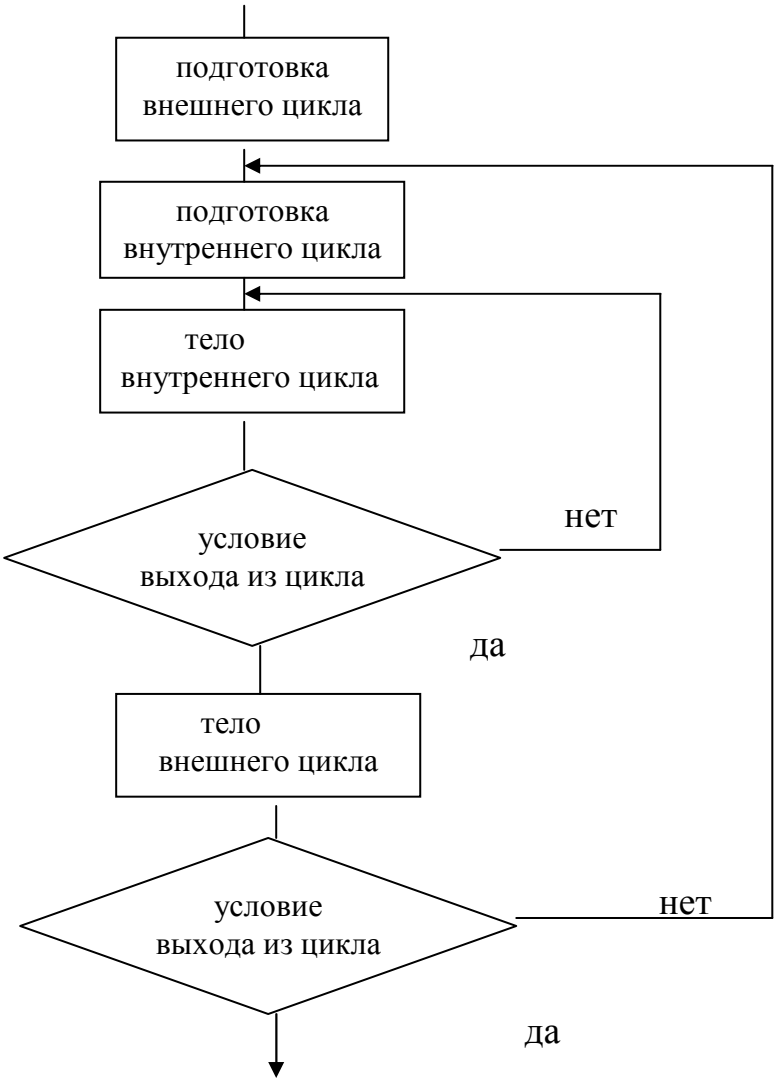


Рис. 8 Схема алгоритма сложного цикла глубиной два

Если несколько усложнить условия предыдущей задачи, то для ее решения придется использовать алгоритм сложного цикла.

Задание № 8. *Найти и вывести все простые числа от 1 до 1000 (сложный цикл).*

```
Program zadacha_8;
Var
I, j, n: integer;
Begin
{Выведем все простые числа от 1 до 1000};
n := 0;
For i := 1 to 1000   do begin              {начало внешнего цикла};
For j := 2 to i - 1   do    If i Mod j = 0 then goto M1; {внутренний цикл};
n: = n + 1 ;
Writeln (‘Организуем вывод простых чисел’ , i);
```

```
If n Mod 16 = 0 then      Writeln();   {Вывод на экран по 16 чисел в строке};
    M1: End;                          {Завершение внешнего цикла};
    Writeln (' Количество простых чисел на интервале:' , n);
     End.
```

Задание № 9. *Составить схему алгоритма (рис.9)* и *вывести таблицу значений функции Z, зависящей от двух переменных x, y, которые изменяются пошагово независимо друг от друга, каждая на своем интервале:*

$$Z = \frac{\sin xy}{\sqrt[2]{x^2 - y^2}}, \text{ если } a \geq x \geq b,$$

$$\Delta x = m; c \geq y \leq d, \Delta y = h$$

Для того чтобы найти значения функции при всевозможных сочетаниях значений аргументов, нужно для каждого значения одной переменной (например, x) перебирать все значения второй переменной (например, y). Т.е. необходимо организовать сложный цикл с двумя независимыми параметрами.

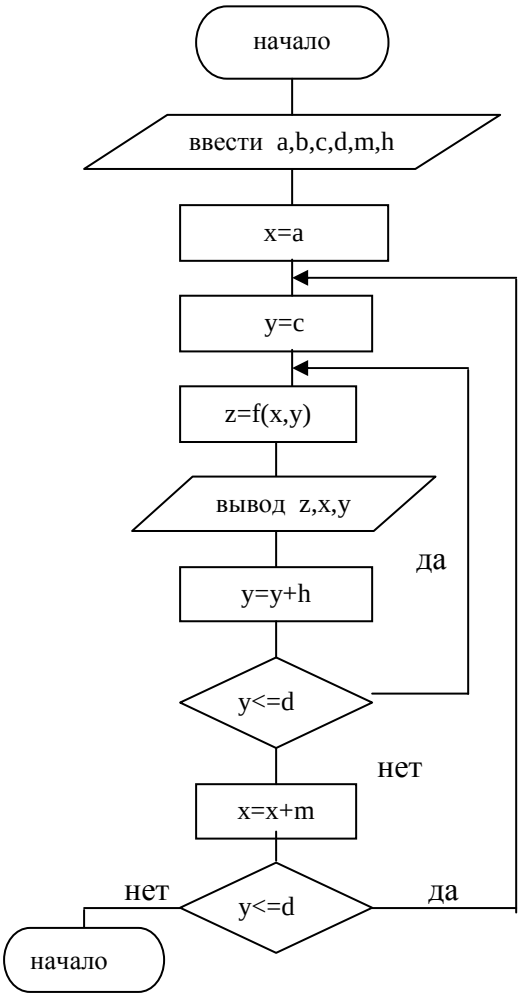


Рис. 9 Схема алгоритма к заданию №9.

```
Program zadacha_9;
    { сложный цикл};
Var
Begin
Writeln ('Введите значения a, b, c, d, m, h');
Readln (a,b,c,d,m,h);
{Выводим заголовок таблицы:};
Writeln(' Z          X          Y ');
  For x = a to b  do begin
     For y = c to d   do begin
z:=Sin(x*y)/Sqrt(sqr(x)-
sqr(y));
Writeln (z, x, y);
y:= y+h-1;
end;
        x:=x+m-1;
end;
End.
```

Самостоятельная работа № 11

1. Набрать, отладить и выполнить программы, реализующие условные алгоритмы Вашего индивидуального задания.
2. Составить блок-схему.
3. Проанализировать работу операторов, пользуясь отладочными режимами.
4. Составить краткий конспект. Защитить работу.

Самостоятельная работа состоит из трех задач. Студент выбирает из списка заданий свой индивидуальный вариант и выполняет его.

Вариант № 1

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{1}{i^2+2i}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше t (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{(-1)^{i+1}}{i^2}$$

Задание № 3.

Дано натуральное число *n*. Определить количество цифр в этом числе.

Вариант №2

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{1}{i^2+1}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше t (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{1}{i(i+1)}$$

Задание № 3.

Дано натуральное число *n*. Переставить первую и последнюю цифры

этого числа.

Вариант №3

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{(-1)i^2}{i!}$$

Задание № 2.

Дано натуральное число *n*. Определить произведение всех цифр этого числа.

Задание № 3.

Вводится последовательность ненулевых чисел, завершаемая нулем. Определить максимальное положительное число в данной последовательности.

Вариант № 4

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{1}{(2i)^2}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше t (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{(-2)i}{i!}$$

Задание № 3.

Дано натуральное число *n*. Определить сумму цифр этого числа.

Вариант №5

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{(-1)^i}{(2i+1)^2}$$

Задание № 2.

Дано натуральное число *n*. Прибавить по единице в старший и младший разряды этого числа.

Задание № 3.

Вводится последовательность ненулевых чисел, завершаемая нулем. Определить максимальное отрицательное число в данной последовательности.

Вариант № 6

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{1}{(2i+1)i}$$

Задание № 2.

Дано натуральное число *n*. Определить, является ли оно простым.

Задание № 3.

Вводится последовательность ненулевых чисел, завершаемая нулем. Определить минимальное отрицательное число в данной последовательности.

Вариант № 7

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{(-1)}{i(i+1)}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше t (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{1}{i^3}$$

Задание № 3.

Дано натуральное число n ($n>2$). Прибавить единицу к старшему разряду числа.

Вариант № 8

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{i+1}{i!}$$

Задание № 2.

Дано натуральное число *n*. Определить входит ли цифра *m* в запись числа *n*.

Задание № 3.

Дано натуральное число *n*. Вывести такие натуральные неотрицательные числа *a*, *b*, *c*, *d*, что $n=a^2+b^2+c^2+d^2$ (это справедливо для любого *n*, согласно теореме Лагранжа).

Вариант № 9

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{2+i}{i!}$$

Задание № 2.

Определить сумму ряда с заданной точностью *t* ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше *t* (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{i}{i^2+i+3}$$

Задание № 3.

Даны натуральные числа *n* и *m*. Вывести все простые числа в диапазоне от *n* до *m*.

Вариант № 10

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{(-1)i^2}{2(i^2+4)}$$

Задание № 2.

Определить сумму ряда с заданной точностью *t* ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше *t* (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{(1+i)^2}{i!}$$

Задание № 3.

Даны натуральные числа *m* и *n*. Получить произведение всех простых натуральных чисел в диапазоне от *m* до *n*.

Вариант № 11

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{(-1)i}{(i+1)^2}$$

Задание № 2.

Вводится последовательность ненулевых чисел, завершаемая нулем. Определить, сколько раз в этой последовательности меняется знак чисел.

Задание № 3.

Даны натуральные числа *m* и *n*. Получить сумму всех простых натуральных чисел в диапазоне от *m* до *n*.

Вариант № 12

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{i}{i^2+2}$$

Задание № 2.

Дано натуральное число *n*. Вывести все совершенные числа, меньшие *n* (число является совершенным, если оно равно сумме своих делителей кроме себя самого).

Задание № 3.

Дано натуральное число *n*. Вывести *n* первых простых чисел.

Вариант № 13

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{i+1}{3i-2}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше t (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{(-1)(i+1)}{i!}$$

Задание № 3.

Дано натуральное число *n*. Найти сумму цифр заданного числа.

Вариант № 14

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{(-1)i}{i^2+i-1}$$

Задание № 2.

Дано натуральное число *n*. Определить разность цифр заданного числа.

Задание № 3.

Числа Фибоначчи f_n определяются следующим образом: $f_0=f_1=1$; $f_n=f_{n-1}+f_{n-2}$. Определить сумму чисел Фибоначчи, не превосходящих некоторого заданного числа *s*.

Вариант № 15

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{i}{2i+1}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше t (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{1}{i^2}$$

Задание № 3.

Дано натуральное число *n*. Приписать по единице справа и слева от этого числа.

Вариант № 16

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{i+1}{i+2}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых, необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше t (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** … (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty}\frac{1}{i(i+1)(i+3)}$$

Задание № 3.

Дано натуральное число *n*. Поменять порядок цифр в этом числе на обратный.

Вариант № 17

Задание № 1.

Вычислить выражения, используя для организации цикла операторы ***For*** (значения переменной *n* задавать с клавиатуры):

$$1+\sum_{i=1}^{n}\frac{(-1)i^2}{3i+2}$$

Задание № 2.

Определить сумму ряда с заданной точностью t ($t>0$) и число слагаемых,

необходимых для достижения этой точности. Точность считается достигнутой, если очередное слагаемое по модулю меньше *t* (это и последующее слагаемое не учитываются). Использовать для организации цикла операторы ***Repeat*** или ***While*** (значения буквенных переменных задавать с клавиатуры):

$$\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{2i-1}$$

Задание № 3.

Дано натуральное число *n*. Поменять местами старший и младший разряд числа.

Лабораторная работа № 9. Работа с массивами.

Цель работы:

1. Изучение приемов программирования с использованием массивов.
2. Закрепление навыков работы в отладочных режимах среды Turbo Pascal.

Массивом называют совокупность данных *одного типа*, обозначаемую одним именем. В зависимости от типа данных массивы могут быть как числовыми, так и текстовыми. При работе с массивами в ЭВМ под каждый элемент массива отводится ячейка памяти, обращение к которой осуществляется с помощью имени массива с индексом, например A[15]. Положение элемента в массиве определяется индексами: одним - для одномерных массивов, двумя - для двумерных (матриц) и т.д. В языке ***Turbo Pascal*** допускаются массивы размерностью 255. Максимальное значение каждого индекса не должно превышать 32767.

Имя массива образуется аналогично имени простой переменной. Индексы заключаются в квадратные скобки и разделяются запятой, если массив не одномерный. В качестве индексов могут использоваться числа, переменные или арифметические выражения, значения которых автоматически округляются до целого. Если индексы не числовые, то их значения должны быть определены заранее.

Примеры обозначения в языке ***Turbo Pascal*** элементов массивов:

***Mas*[33], *Mas*[*i*], *Mas*[*i* + 4]** - для одномерного массива;
***Mas*[12,3], *Mas*[*i*,*j*], *Mas*[*i*+2,*j*+3]** - для двумерных массивов.

Массив должен быть объявлен соответствующим оператором в области описания переменных ***Var***.

В операторе указываются имена массивов и в квадратных скобках верхние и нижние границы изменения индексов, которые должны быть целыми положительными числами или переменными, значение которых определено в программе ранее.

Например, ***Mas array* [5 .. 50] of *Real*;** - оператор описывает одномерный массив, имя которого ***Mas***, а индексы могут принимать значения от 5 до

50, т.о. под этот массив выделяется 46 ячеек памяти.

Значение нижней границы индексов может быть опущено, и тогда по умолчанию оно принимается равным единице, например:

***Massiv1 array* [15] *of Real*;**

***Massiv2 array* [5,8] *of Real*;** - операторы описывают два массива:

- Одномерный массив с именем ***Massiv1*** и индексы могут принимать значения от 1 до 15, т.о. зарезервировано 15 ячеек;

- двумерный массив(матрицу) с именем ***Massiv2***, при этом индекс строки может принимать значения от 1 до 5, а индекс столбца - от 1 до 8.

При обозначении двумерных массивов индекс строки стоит на первом месте, индекс столбца - на втором**.**

В языке ***Turbo Pascal*** обработка массивов осуществляется поэлементно, в том числе и ввод - вывод массива. Если массив содержит всего несколько элементов, то задать их значения можно с помощью операторов присваивания:

***Massiv1 array* [4] *of Real*;**
***Massiv1* [1]:=0.25; *Massiv1* [2]:=0.12;**

***Massiv1* [3]:=0.35; *Massiv1* [4]:=0.28;**

или с помощью оператора ввода:

***Massiv1 array* [4] *of Real*;**
Readln* (*Massiv1[1], Massiv1[2], Massiv1[3], Massiv1[4]);

Аналогичным образом осуществляется и вывод массива:
Writeln* (*Massiv1[1], Massiv1[2], Massiv1[3], Massiv1[4]);

В том случае, когда массив содержит много элементов и перечисление их при вводе - выводе становится неудобным**, организуется цикл.**

Далее приведены фрагменты программ - ввода и вывода значений элементов одномерного массива:

```
Program primer_1;
{ Ввод  массива с помощь  оператора " Readln"};
Var
    Mas1 array [n] of real;
    n, i: integer;
Begin
    Writeln ('Введите кол-во элементов в массиве');
     Readln (n);
For i:=1 to n do begin
            Writeln('Введите значение элемента', i);
             Readln (Mas1[i]);  End;
End.
```

```
Program primer_2;
{ Ввод  массива с помощью датчика случайных чисел };
Var
     Mas1 array [n] of real;
     n, i: integer;
Begin
     Randomize;
     Writeln (‘Введите кол-во элементов в массиве’);
     Readln (n);
For i:=1 to n do Mas1[i] :=int(Random(250));
End.
```

```
Program primer_3;
{ Ввод  массива с помощь  оператора " Readln"};
Var
     Mas1 array [n] of real;
     n, i: integer;
Begin
     Writeln (‘Введите кол-во элементов в массиве’);
     Readln (n);
For i:=1 to n do begin
                Writeln(‘Введите значение элемента’, i);
                 Readln (Mas1[i]);  End;
{ Вывод  массива  экран с помощь  оператора " Writeln"};
For i:=1 to n do Writeln(‘Mas[’, i, ‘]=’, Mas1[i]);
End.
```

Задание №1. *Нахождение максимального элемента одномерного массива.*

```
Program zadanie_1;
{Ввод массива с помощью случайных чисел};
Var
    Ms array [n] of integer;
   n, i, max, max_i: integer;
Begin
   Writeln (‘Введите кол-во элементов в массиве’);
   Readln (n);
   Randomize;
   For i:= 1 to n do Ms[i]:=int(Random(250));
  {Нахождение максимального элемента массива};
  max:= Ms[1];
```

```
 max_i:= 1;
 For i:=2 to n do begin
                If max < Ms[i] then begin
                                max:= Ms[i];   max_i:= i; end;
                 end;
{Выводим значение  максимального 'элемента массива и его  номер };
Writeln (‘Самый большой элемент’, max,’находится по адресу ’,max_i);
End.
```

Для ввода-вывода двумерных массивов - матриц, организуется сложный (вложенный - глубиной два) цикл:

```
Program primer_4;
{Ввод двумерного массива с помощью оператора Readln};
Var
   Ms array [4, 7] of real;
   i, j: integer;
Begin
    For i:=1 to 4 do begin
            For j:=1 to 7 do begin
                Writeln (‘Введите элемент массива’);
                  Readln (i, j);  end;
            End;
End.
```

```
Program primer_5;
{'Ввод массива Ms(i, j)  с помощью датчика случайных чисел};
Var
   Ms array [i, j] of real;
  i, j, k, m: integer;
Begin
Randomize;
Writeln (‘Введите кол-во строк и кол-во элементов в строке’);
Readln( k, m);
For i:=1 to k do begin
            For j:=1 to m do Ms[i,j]:= Random(250);  End;
{ Вывод  массива  на экран с помощь  оператора " Writeln"};
For i:=1 to k do begin
       For j:=1 to m do  Writeln(‘Mas[’, i, ‘,’, j,‘]=’, Mas1[i, j]);
          End;
End.
```

Задание №2. *Сформировать одномерный массив из максимальных элементов строк матрицы* ***Ms[22,5]****. В свою очередь матрицу* ***Ms*** *получить с помощью датчика случайных чисел.*

```
Program  zaganie_2;
{Ввод массива Ms[22,5] с помощью датчика случайных чисел};
Var
Ms array [22,5] of real;
Ms_1 array [22] of real;
i, j : integer;
max: real;
Begin
Randomize;
For i:=1 to 22 do begin
For i:=1 to 22 do begin
                For j:=1 to 5 do Ms[i,j] := random(200);    end;
{Находим максимальное значение элемента в строке};
For i:=1 to 22 do begin
            max:= Ms[i,1];
             For j:=1 to 5 do begin
                          If max<Ms[i,j] then max:= Ms[i,j]; end;
            Ms_1[i] := max;    end;
{ Вывод  массива  на экран с помощь  оператора " Writeln "};
      For i:=1 to 22 do Writeln ('Ms_1[', I, ']=', Ms_1[i]);
End.
```

Задание №3. *Заполнить двумерный массив случайными числами. Все элементы 1-ой строки прибавить ко всем элементам других строк массива. Полученный результат вывести на экран.*

```
Program  zaganie_3;
{Ввод массива Ms[n,k] с помощью датчика случайных чисел};
Var
Ms array [n,k] of real;
i, j,n,k : integer;
Begin
Randomize;
Writeln ('Введите кол-во строк и кол-во элементов в строке');
Readln (n,k);
For i:=1 to n do begin
                For j:=1 to k do Ms[i,j] := random(200);    end;
{Все элементы 1-й строки прибавляем ко всем элементам };
For i:=2 to n do begin
                        For j:=1 to k do Ms[i,j] :=  Ms[i,j]+ Ms[1,k];  end;
```

```
{ Вывод  массива  на экран с помощь  оператора " Writeln "};
      For i:=1 to n do  begin
                  For j:=1 to k do Writeln ('Ms_1[', I, ']=', Ms_1[i]);  end;
End.
```

Самостоятельная работа № 12.

1. Набрать, отладить и выполнить программы, реализующие условные алгоритмы Вашего индивидуального задания.
2. Составить блок-схему.
3. Проанализировать работу операторов, пользуясь отладочными режимами.
4. Составить краткий конспект. Защитить работу.

Самостоятельная работа состоит из двух задач. Студент выбирает из списка заданий свой индивидуальный вариант и выполняет его.

Вариант №1

Задание 1.

Исходный массив элементов получить с помощью датчика случайных чисел. Определить сумму и количество положительных элементов массива *X[n]*. Вывести исходный массив и полученные значения. Количество элементов задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Возвести ее во вторую степень. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант №2

Задание 1.

Определить среднее арифметическое значение элементов массива *X[n]*. Вывести исходный массив и полученное значение. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Исходную матрицу получить с помощью датчика случайных чисел. Элементы последней строки прибавить ко всем элементам других строк. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант №3

Задание 1.

Определить наибольший элемент массива *X[n]*. Вывести исходный массив, этот элемент и его порядковый номер. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел $X[n, m]$. Определить номер строки матрицы, сумма элементов которой минимальна. Вывести исходную матрицу, минимальную сумму и номер соответствующей строки. Значения буквенных переменных задавать с клавиатуры.

Вариант №4

Задание 1.

Переписать элементы массива $X[n]$ в массив $Y[n]$ в обратном порядке. Вывести оба массива. Определить наименьший элемент массива $Y[n]$ и его месторасположение. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел $X[n, m]$. Определить номер строки матрицы, сумма элементов которой максимальна. Вывести исходную матрицу, максимальную сумму и номер соответствующей строки. Значения буквенных переменных задавать с клавиатуры.

Вариант №5

Задание 1.

Определить наименьший элемент массива $X[n]$. Вывести исходный массив, этот элемент и его порядковый номер. Вывести полученный массив. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел $X[n, m]$. Получить новый массив $Y[n]$, состоящий из суммы строк исходной матрицы $X[n, m]$. Вывести исходную матрицу и полученный массив. Значения буквенных переменных задавать с клавиатуры.

Вариант № 6

Задание 1.

Переписать все нечетные элементы исходного массива $X[n]$ в массив $Y[n]$. Определить сумму элементов массива с нечетными номерами. Вывести полученную сумму. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел $X[n, m]$. Получить новый массив $Y[n]$, состоящий из минимальных элементов строки исходной матрицы $X[n, m]$. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 7

Задание 1.

Определить сумму элементов массива $X[n]$ с четными номерами. Вывести массив и полученную сумму. Исходный массив элементов получить с помощью датчика случайных чисел. Возвести в квадрат все элементы исходного массива $X[n]$ и получить новые значения. Вывести полученный массив. Значения

буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новый массив *Y[n]*, состоящий из среднеарифметических значений элементов строки исходной матрицы *X[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 8.

Задание 1.

Сформировать массив *Y* из элементов массива *X[n]*, расположив сначала положительные, а потом отрицательные элементы. Вывести оба массива. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новый массив *Y[n]*, состоящий из суммы квадратов элементов строки исходной матрицы *X[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 9

Задание 1.

Сформировать массив *Y* из элементов массива *X[n]*, изменив значение элементов в соответствии с формулой $(X[n]+n)^2$. Вывести оба массива. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Найти наибольший элемент массива, начиная со следующего все элементы массива увеличить на наибольший элемент. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 10

Задание 1.

Определить сумму отрицательных и количество положительных элементов массива *X[n]*. Вывести массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Найти наибольший и наименьший элементы массива, начиная со следующего за наименьшим, все элементы увеличить на наибольшее значение элемента, а полученную сумму умножить на наименьшее значение. Вывести полученную матрицу и адреса найденных значений. Значения буквенных переменных задавать с клавиатуры.

Вариант № 11

Задание 1.

Определить сумму элементов массива *X[n]* с нечетными номерами.

Вывести массив и полученную сумму. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новую матрицу, элементы строк которой больше элементов исходной матрицы на величину максимального элемента соответствующей строки исходной матрицы. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 12

Задание 1.

Определить значение и номер элемента массива *X[n]*, наиболее близкого к среднему арифметическому всех элементов массива. Вывести массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Все элементы строки умножить на среднеарифметическое значение данной строки и полученные значения поместить в массив *Y[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 13

Задание 1.

Дан массив целых чисел *X[n]*. Найти наибольший элемент в массиве, начиная с этого элемента все остальные элементы умножить на наибольший, а полученный результат возвести в квадрат. Вывести полученный массив. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Каждое значение элемента матрицы возвести в квадрат, к полученному результату прибавить его адрес. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 14

Задание 1.

Дан массив целых чисел *X[n]*. Определить количество и сумму положительных элементов массива. Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Вычесть поэлементно из каждого столбца, кроме *i*-го, *i*-й столбец. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 15

Задание 1.

Определить количество и произведение четных элементов массива *X(n).* Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Вычесть поэлементно из каждой строки, кроме *i*-й, *i*-ую строку. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 16

Задание 1.

Определить количество элементов массива *X[n],* заканчивающихся цифрами 5 или 7. Найти сумму элементов, заканчивающихся цифрами 5, и разность элементов, заканчивающихся цифрами 7. Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Даны матрицы целых чисел *X[n, m]* и *Y[n, m]*. Получить их произведение, а результат поместить *X[n, m]*. Вывести исходные и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 17

Задание 1.

Определить для массива *X[n]* алгебраические суммы четных и нечетных элементов массива. Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Определить максимальный и минимальный элементы матрицы и поменять их местами. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Лабораторная работа № 10. Строковый тип данных.

Цель работы:

1.Изучение приемов программирования с использованием строковых данных.

2.Приобретение практических навыков в работе со строковыми данными.

Описание и ввод строковых данных

Ранее упоминалось, что в языке Turbo Pascal существует строковый тип данных для обработки последовательности символов. Данными строкового

типа являются строковые константы и строковые переменные.

Строковая **константа** представляет собой произвольную последовательность символов, заключенную в одинарные кавычки, длиной до 32567 символов, например, *‘ Hello’, ‘Добрый день’*.

Строковые **переменные** бывают переменной или фиксированной длины. Строка переменной длины (***String)*** представляет собой последовательность длиной до 32567 символов из таблицы ASCII. В памяти под такую символьную переменную отводится количество байт равное количеству символов переменной плюс 4. Объявить строковый тип переменной длины можно с помощью оператора объявления типа данных:

Var Str: String;

Строка фиксированной длины ***Str1:String [n];*** представляет собой строку длиной ***n*** символов. В памяти под такую символьную переменную отводится ***n*** байт. Описать символьную переменную фиксированной длины можно таким образом:

```
Var Hello :String[12];
Hello := ‘Привет – Hello’;
Writeln(‘ Результат:’, Hello);
```

На экран будет выведено: Результат: ***Привет - Hello***

Работа со строками. Строковые операции.

Строковые выражения используются в различных операторах языка ***Turbo Pascal***: присваивания, условного перехода, вывода и т.д.

Строковое выражение может содержать строковые константы, строковые переменные, вызов функций и строковые операции.

1. Операция "+" (конкатенация) предназначена для объединения строк. Результат операции имеет строковый тип. Например, после выполнения фрагмента программы***:***

```
L:= ‘MOSCOW’;
AGE:= ‘We’ + ‘ live in ‘ +  L;
```

строковая переменная ***AGE*** примет значение:

We live in MOSCOW.

2. Операции сравнения (=, <>, <, >, <=, >=). Сравнение двух строк выполняется слева направо с учетом кодов ASCII. Т.е. сравниваются сначала коды первых символов, затем вторых и т.д. Результат операций сравнения имеет логический тип, то есть принимает значения **ДА** или **НЕТ**, например,

‘A’ < ‘B’ (результат ДА)

‘RA’ > ‘RR’ (результат НЕТ)

‘002’> ‘12’ (результат ДА)

Если две строки имеют различную длину, но их начальные символы совпадают, включая последний символ более короткой строки, то короткая строка считается меньшей, например:

‘12.0’ > ‘12’ (результат ДА).

Строки считаются равными тогда и только тогда, когда имеют одинаковую длину и одинаковую последовательность символов, например:

'TURBO' = 'TURBO' (результат ДА)

*'**TURBO' = ' TRUBO** '* (результат НЕТ).

Строковые функции и операторы

Приведем наиболее часто употребляемые строковые функции:

– ***Val(St)*** – преобразует строковое выражение в его численное представление.

– ***Length(St)*** в качестве результата дает целое число, равное длине строкового выражения ***St***.

– ***Copy(St, n, m)*** в качестве результата дает фрагмент строки ***St***, длиной m, начиная с позиции n.

– ***Str(числовое выражение)*** - преобразует числовое выражение в символьное. Если его значение положительно, то к полученной строке слева добавляется пробел.

– ***Delete(St,n,m)*** удаляет из строки ***St*** подстроку длиной ***m***, начиная с позиции ***n.***

– ***Pos(St,'stroka')*** производит поиск подстроки в строке и выдает № позиции, или **0** – если подсторка не найдена.

– ***Insert(St, 'stroka',n)*** добавляет подстроку ***'stroka'*** в строку ***St,*** начиная с ***n.***

– ***Chr(****integer)* определяет по коду ASCII символ.

– ***Ord****(string)* вычисляет порядковый номер символа.

Задание № 1. *Дан текст. Удалить из текста повторяющиеся символы.*

```
Program zadacha_1;
Var
Slovo: string;
dl, i: integer;
Begin
Slovo :=' ббааабоччкаа';    { Исходный текст };
Writeln ( Slovo);
dl := Length(Slovo);          {Определяем длину строки};
 i:=1;
{Сравнение 2-х соседних символов и при равенстве удаляем 1 символ};
{При удалении 1-го символа уменьшаем длину исходной строки на 1};
while  i<dl do begin
            if copy(Slovo,i ,1)= copy(Slovo, i+1, 1) then begin
                                                  delete(Slovo, i, 1);
```

```
                                                dl := dl-1;
                                               end;
            i:= i+1;
           end;
Writeln (Slovo);
End.
```

Результат:

```
бабочка
```

Задание № 2. *Исходный текст вводится с клавиатуры. Найти наибольшее количество цифр, идущих подряд.*

```
Program zadacha_2;
Var
stroka: string;
     l, m, i, dl: integer;
{l- максимальное количество цифр в тексте, идущих подряд };
{m- счетчик идущих подряд цифр в группе };
Begin
    Writeln ('Введите исходный текст');
   Readln (stroka);
  Writeln ('Исходный текст:', stroka);
 dl: = Length(stroka);   {длина строки};
l:=0;   m:=0;
{Анализируем всю строку};
for i:=1 to dl do  begin
                    {Проверяем. Является ли текущий символ цифрой};
                  if (copy(stroka,I,1) = i) and (i>=0) and (i<=9) then m:=m+1
                    else if l <m then begin
{Если текущий символ - не цифра, то начинаем считать сначала};
{Проверяем, если в группе больше цифр, то сохраняем это значение в l};
                                                l:=m;
                                               m:= 0;   end;
                   end;
Writeln ('количество цифр в тексте =', l);
End.
```

Результат:

```
Исходный текст:
asd123d343434dd34vv6876543321321rtr555rtyhgf6667u7uyuy888888123
Наибольшее количество цифр в тексте = 13
```

Пояснение к программе

Данная задача - есть задача нахождения максимума, поэтому под него выделяем ячейку памяти "***l***". После ввода текста просматриваем посимвольно текст и каждый символ проверяем на совпадение с цифрой. При выявлении цифры к текущему значению счетчика ***m*** прибавляется 1. Затем сравниваем текущее значение счетчика с содержимым ячейки "***l***" и если ***m*** больше ***l***, то заменяем значение ячейки "***l***" на большее.

В том случае, когда очередной символ не цифра, мы обнуляем "***m***".

Просматриваем посимвольно текст до его окончания.

Задание №3. *В тексте определить количество предложений, наибольшее количество слов в предложении и номер этого предложения. (При вводе текста в конце предложений ставить точку и один пробел.*)

```
Program zadacha_3;
Var
in_text, vv_text: string;
dl, i, j, end_str, n, n_max, max_slovo: integer;
Begin
vv_text:='В Московском государстве жил был Царь. Был у Царя Стрелец молодец. И был еще  при  Царе хитрый Советник. Был Министр финансов.';
Writeln ( 'Исходный текст -  ',vv_text);
in_text := vv_text;
dl:=Length(in_text); { длина строки };
n:=0;
for i:=1 to dl do begin
    n:= n+1;  { Считаем кол-во предложений};
   end_str:=Pos(in_text, '.'); { Определяем кол-во символов до точки};
   j:=1;

   While j < end_str do begin
{ Предложение разбиваем на слова и их считаем};
                        if Pos(in_text, ' ') <> 0 then begin
                                end_str:= end_str- Pos(in_text, ' ');
                                 in_text := Delete(in_text, 1, Pos(in_text, ' ')+1);
                                 kol_slovo:= kol_slovo+1;   end;
                          end;
         if max_slovo < kol_slovo then begin
                                    max_slovo:=kol_slovo;
                                   n_max:=n;    end;
                 dl:= dl-(kol_slovo+1);
                kol_slovo:=0;
        end;
```

```
Writeln (‘Наибольшее количество слов в предложении = ‘, max_slovo);
Writeln (‘Количество предложений =’,n_max);
End.
```

Файл результата:

```
Исходный текст
в Московском государстве жил был Царь. Был у Царя Стрелец молодец.
И был еще при Царе хитрый Советник. Был Министр финансов.
Наибольшее количество слов в предложении = 7
Количество предложений= 4
```

Группу строк (символов) можно хранить в массиве, например A[i], и обращаться к отдельным строкам по именам A[1],A[2],[3] и т.д. Например, список фамилий в телефонном справочнике можно представить, как одномерный строковый массив.

Для массивов строк допустимы те же имена, что и для числовых массивов. Каждый элемент массива равноценен строковой переменной. Действие всех вышеперечисленных строковых функций распространяется и на элементы массивов.

Задание № 5. *С клавиатуры вводится список фамилий. Распечатать список в алфавитном порядке*.

```
Program zadacha_5;
Var
n, i, j: integer;
mas_fam array [n] of string;
vrem : string;
Begin
{Задача представляет собой  задачу сортировки одномерного массива};
Writeln (‘ Введите кол-во фамилий’);
Readln(n);
For i:=1 to n-1 do begin
                For j:=1 to n-I do begin
{ Если  фамилии не по алфавиту, то меняем  их местами };
                        if mas_fam[i]>mas_fam[j+1] then begin
                                vrem:=mas_fam[i];
                                mas_fam[i]:=mas_fam[j+1];
                                mas_fam[j+1]:=vrem;     end;
                        end;
                end;
For i: = 1 to n do Writeln(mas_fam[i]);
  End.
```

Пояснение к программе

Необходимо ввести дополнительную переменную для промежуточного сохранения значения одной из двух переменных.

Индивидуальное задание по лабораторной работе состоит из трех задач. Далее приведены примеры типовых заданий по программированию задач с использованием строковых данных.

Самостоятельная работа № 13

1. При домашней подготовке составить программы на языке ***Turbo Pascal*** согласно варианту задания.
2. В системе ***Turbo Pascal*** создать файлы программ.
3. Отладить и выполнить программы. Результат вывести на экран.
4. Проанализировать работу операторов и символьных функций.
5. Написать отчет. (Краткий конспект и распечатки программ и результатов.)

Самостоятельная работа состоит из двух задач. Студент выбирает из списка заданий свой индивидуальный вариант и выполняет его.

Вариант № 1.

Задание 1.

Сформировать по строке ***a*** новую строку по правилу: если слово имеет нечетную длину, удалить в нем среднюю букву.

Задание 2.

Дана строка ***a***, состоящая из нескольких слов. Вывести слова, составляющие строку в алфавитном порядке.

Вариант № 2

Задание 1.

Для строки ***a*** подчеркнуть (знак «-» в последующей строке) все русские буквы. Вывести строку с подчеркиванием.

Задание 2.

Вывести исходную строку ***a***, заменив в ней все «,» на «?».

Вариант № 3

Задание 1.

В исходной строке ***a*** определить количество слов и количество букв в нем. Результат вывести на экран.

Задание 2.

В исходной строке ***a*** произвести перестановку по алфавиту всех нечетных слов. Вывести исходную и полученную строки.

Вариант № 4

Задание 1.

В строке ***a*** (на русском языке) произвести замену «:» на символ, введенный с клавиатуры. Вывести полученную строку.

Задание 2.

В исходной строке ***a*** определить количество слов, начинающихся

заданным символом ***x*.** Вывести строку, результат анализа и найденные слова.

Вариант № 5

Задание 1.

В исходной строке ***a*** заменить все четные слова на слово, введенное с клавиатуры. Вывести полученную строку.

Задание 2.

В исходной строке ***a*** произвести перестановку по алфавиту всех четных слов. Вывести исходную и полученную строки.

Вариант № 6

Задание 1.

По исходным строкам ***a*** и ***b*** определить слова, входящие в строку ***b***, но не входящие в строку ***a***, добавить их к концу строки ***a***. Вывести полученную строку.

Задание 2.

Сформировать по строке ***a*** новую строку по правилу: если слово имеет четную длину, удалить в нем первую и последнюю буквы.

Вариант № 7

Задание 1.

В исходной строке ***a***, состоящей из цифр, определить среднее арифметическое цифр, входящих в строку. Вывести результат.

Задание 2.

Для исходной строки ***a***, состоящей из буквенных символов, определить упорядочены ли они по алфавиту. Вывести строку и результат анализа в словесной форме.

Вариант № 8

Задание 1.

В исходной строке ***a*** переставить местами ***n*-*e*** и ***m*-*e*** слова. Вывести полученную строку. Если слов с заданными номерами нет, вывести соответствующий ответ.

Задание 2.

Вывести исходную строку ***a***, заменив в ней все «;» на «!».

Вариант № 9

Задание 1.

В исходной строке ***a*** определить и вывести слова, в которых нет повторяющихся букв.

Задание 2.

Определить для строки ***a*** является ли она симметричной, т.е. читается одинаково слева направо и справа налево. Вывести строку и результат анализа.

Вариант № 10

Задание 1.

Сформировать по исходной строке ***a*** строку ***b*** по правилу: при встрече с строке ***a*** группы символов «***#***» отменять в формируемой строке число символов,

равное количеству символов группы *«#»*. Вывести исходную и полученную строки.

Задание 2.

В исходной строке ***a*** определить число слов, которые содержат хотя бы один символ ***x***. Вывести исходную строку, результат анализа и соответствующие слова.

Вариант № 11

Задание 1.

В исходной строке ***a*** определить и вывести слова, в которых нет повторяющихся букв.

Задание 2.

В исходной строке ***a*** определить количество слов, заканчивающихся заданным символом ***x***. Вывести строку, результат анализа и полученные слова.

Вариант № 12

Задание 1.

В исходной строке ***a*** произвести замену сочетания символов ***x*** на сочетание символов ***y***. Вывести исходную и полученную строки.

Задание 2.

Определить сумму ***ASCII***-кодов символов, составляющих строку ***a***. Вывести полученное значение на экран и записать в строку ***b***, состоящую из символов.

Вариант № 13

Задание 1.

В исходной строке ***a*** определить и вывести слова, перед которыми стоят меньшие по длине слова.

Задание 2

Сформировать по исходной строке ***a*** новую строку ***b*** по правилу: в каждом слове перенести первую букву в конец слова.

Вариант № 14

Задание 1.

В исходной строке ***a*** определить количество слов, содержащих ровно ***n*** символов строки ***x***.

Задание 2.

Сформировать по строке ***a*** новую строку по правилу: если слово имеет четную длину, удалить в нем вторую букву.

Вариант № 15

Задание 1.

В исходной строке ***a*** определить и вывести слова, в которых нет повторяющихся букв.

Задание 2.

Сформировать по строке ***a*** новую строку по правилу: если слово имеет нечетную длину, удалить в нем все буквы «**а**».

Вариант № 16

Задание 1.

В исходной строке ***a*** определить и вывести слова, которые встречаются в строке по одному разу.

Задание 2.

Сформировать по строке ***a*** новую строку по правилу: если слово имеет нечетную длину, то произвести перестановку нечетных слов, например: меняем местами 2-е и 4-е слово, 6-е и 8-е слово и т.д.

Вариант № 17

Задание 1.

В исходной строке ***a*** определить сколько в ней находится предложений и сколько слов в каждом предложении. Результат вывести на экран.

Задание 2.

Вывести исходную строку ***a***, удалив из нее лишние (следующие подряд) пробелы.

Лабораторная работа №11. ***Работа в графическом режиме.***

<u>Цель работы:</u>

1. Изучение приемов программирования с использованием графического режима.
2. Приобретение практических навыков работы в графическом режиме.

Для воспроизведения графики компьютер снабжен специальными аппаратными средствами. К ним относятся монитор и специальное устройство - видеоадаптер (видеокарта), выполняющее роль переводчика между памятью и экраном. Видеоадаптер вместе с монитором образуют видеоподсистему.

Видеоподсистемы работают в двух видеорежимах: текстовом или графическом. В текстовом режиме экран монитора разбивается на отдельные символьные позиции, в каждой из которых может выводиться только один символ.

В графическом режиме для каждой точки изображения, называемой пикселем, отводится от одного (монохромный режим) до 24-бит (цветной). В этом режиме имеется доступ к каждой точке изображения. Любое изображение можно представить в виде множества мельчайших точек, каждой из которых сопоставлены две координаты и номер цвета. Полученный числовой набор, называемый ***растром***, более или менее точно опишет изображение. Графические режимы используются для формирования рисунков.

В программировании используется такая характеристика как ***разрешение***. Для графических режимов - это количество доступных точек на экране, для текстовых - количество символов в строке. ***Разрешение экрана*** является одним из важнейших параметров видеоподсистемы. Чем оно выше, тем больше информации можно отобразить на экране.

Количество различных цветов (ц*ветовое разрешение)*, доступных для

раскрашивания изображений - другое важное свойство графического режима. Базовая палитра IBM - совместимых ЭВМ включает 16 стандартных цветов. Современные персональные компьютеры комплектуются дисплеями способными отображать палитры от 256 до 16 млн. цветов.

В графическом режиме каждый пиксель определяется цветом и своими координатами - положением относительно левого верхнего угла экрана, который, в свою очередь, имеет координаты 0,0. Программист может управлять цветом любого пикселя, что позволяет формировать на экране любые изображения, в том числе рисунки, графики, чертежи, символы.

В текстовых режимах можно задавать координаты символа, определяя положение курсора, относительно левого верхнего угла экрана (1,1), цвет символа (цвет переднего плана) и цвет фона (цвет заднего плана).

Стандартный модуль GRAPH содержит библиотеку из более чем 50 графических программ. В нем поддерживается несколько видов закрашивания, типов линий и шрифтов, размер которых можно изменять.

Для переключения экрана в графический режим необходимо выполнить процедуру:

InitGraph (var Graphdriver : integer; var GraphMode : integer; PathToDriver : string);

По этой процедуре выполняются:

− поиск в каталоге ***PathToDriver*** файла драйвера графической карты с номером ***GraphDriver***;

− загрузка драйвера в оперативную память;

− установка указанного графического режима ***GraphMode***.

Если имя каталога не указано (параметр ***PathToDriver*** представляет собой пустую строку ''), то файл ищется в текущем каталоге.

Если ***GraphDriver = 0***, то происходит автоматическое определение графической среды, а переменным ***GraphDriver*** и ***GraphMode*** присваиваются величины, определяющие номер драйвера и номер режима графического экрана.

Номера карт (графических драйверов) и режимов могут быть выражены с помощью следующих обозначений:

Detect=0 - автоматическое распознавание графической карты;
CGA=1 - карта CGA;
MCGA=2 - карта MCGA;
EGA=3 - карта EGA;
EGA64=4 - карта EGA64;
EGAMono=5 - карта EGAMono;
Reserved=6 - зарезервировано (не используется);
HercMono=7 - карта Hercules;
ATT400=8 - карта ATT400;
VGA=9 - карта VGA;
PC3270=10 - карта PC3270,

а также используются некоторые символы режима:

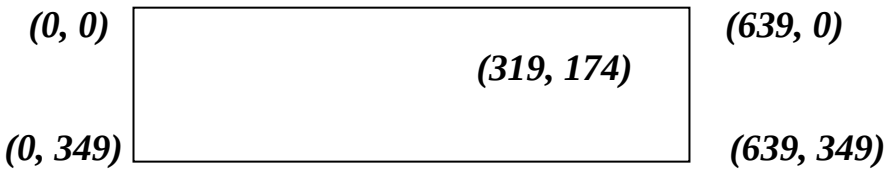
EGALo=0 - 620x200 пикселей, 16 цветов, 4 страницы;

EGAHi=1 - 640x350 пикселей, 16 цветов, 2 страницы.

Пиксель - это единичный элемент экрана. Количеством пикселей по горизонтали и вертикали определяется размер графического экрана. Синонимом пикселя является понятие "***точка***".

Для перехода из графического режима в текстовый используется процедура ***CloseGraph*** без параметров, которая устанавливает текстовый режим и очищает память, автоматически выделенную программе. Это выполняется подсистемой управления графикой.

Для того чтобы с помощью операторов, строящих изображение, можно было обращаться к заданным точкам экрана, необходимо иметь возможность однозначно их идентифицировать. Положение точки (пикселя) задается с помощью двух координат ***X, Y***, где ***X*** - номер позиции по горизонтали, ***Y*** - номер позиции по вертикале. По соглашению верхний левый угол экрана имеет координаты ***(0,0).*** Координата ***X*** увеличивается при перемещении вправо, а координата ***Y*** - при перемещении вниз. Таким образом, координаты каждого из четырех углов и конкретной точки, например, середины экрана, могут выглядеть следующим образом:



На данном рисунке рассмотрен графический режим с матрицей экрана 640x350 пикселей. В некоторых других режимах эта матрица может быть иной, например, 320x200 или 640x200 и так далее.

Определение координат правого нижнего угла экрана выполняется по функциям: ***GetMaxX, GetMaxY***.

Синтаксис: ***GetMaxX : integer; GetMaxY : integer.***

Графические объекты могут быть черно-белыми или цветными. Черно-белые рисунки выполняются обычно в системах с графическими картами, не учитывающими цвет, например, с картой ***Hercules***. Если возможен вывод в цвете (например, карта ***EGA*** или ***CGA***), то при установке графического режима можно определить его параметры и тем самым доступную цветовую гамму. Цвета гаммы (она отсутствует для карты ***Hercules***) пронумерованы от ***0*** до ***GetMaxColor.*** Цвета могут быть фиксированными (для карты ***CGA***), либо произвольно изменяемыми (для карт ***EGA*** и ***VGA***). Если цвета могут изменяться, то присвоение выбранного цвета позиции цветовой гаммы производится с помощью процедуры ***SetPalette***.

Синтаксис: ***SetPalette (ColorNum, Color : word);***

Идентификатор цвета ***Color*** присваивается позиции ***ColorNum*** цветовой гаммы. Изменение цвета на экране будет обнаружено после выполнения процедуры ***SetPalette***.

Идентификаторы цветов могут быть выражены следующим образом:

Black=0 (черный);	***DarkGray=8*** (темно-серый);
Blue=1 (синий);	***LightBlue=9*** (светло-синий);
Green=2 (зеленый);	***LightGreen=10*** (светло-зелен.);
Cyan=3 (голубой);	***LightCyan=11*** (светло-голуб.);
Red=4 (красный);	***LightRed=12*** (светло-красн.);
Magenta=5 (лиловый);	***LightMagenta=13*** (светло-лилов.);
Brown=6 (коричневый);	***Yellow=14*** (желтый);
LightGray=7 (светло-серый);	***White=15*** (белый).

Выбор номера цвета для изображения объектов обеспечивает процедура ***SetColor***.

*Синтаксис**:*** ***SetColor(N : word);***

При выполнении этой процедуры объекты будут рисоваться цветом, связанным аппаратно или программно с позицией N цветовой гаммы. Программное присвоение этой позиции другого цвета немедленно изменяет цвет рассматриваемых объектов.

Процедура ***SetBkColor*** изменяет цвет фона на такой цвет, который соответствует позиции N текущей цветовой гаммы.

Синтаксис: ***SetBkColor(N : word).***

Рисование графических примитивов

Рисование точки (пикселя) с координатами (***X, Y***) цветом с номером ***Color*** выполняется процедурой ***PutPixel(X, Y: integer; Color: word).***

Рисование отрезков прямых линий можно выполнить одной из следующих трех процедур:

Line(X1, Y1: integer; X2, Y2: integer) – рисование отрезка прямой линии, соединяющего точки с координатами (***X1,Y1***) и (***X2,Y2***);

LineRel(dX, dY: integer) - рисование отрезка прямой линии от текущего положения графического курсора на расстояние ***dX*** по горизонтали и ***dY*** по вертикали;

LineTo(X, Y: integer) - рисование отрезка прямой линии от текущего положения графического курсора к точке, имеющей координаты (***X, Y***).

После установления графического режима по умолчанию графическим окном будет весь экран. Левый верхний угол графического экрана имеет координаты (0,0), а правый нижний (***GetMaxX, GetMaxY***).

Графическое окно можно установить с помощью процедуры:

SetViewPort(X1, Y1: integer; X2, Y2: integer; Clip :boolean);

Процедура задает графическое окно в виде прямоугольника с координатами противоположных углов (***X1, Y1***) и (***X2, Y2***). ***CLIP*** определяет, должны ли обрезаться выходящие за пределы окна части рисунка: если этот

параметр задается константой ***ClipOff*** (или False), то не обрезать, если же константой **ClipOn** (или ***True)***, то обрезать.

Все графические операции выполняются в текущем графическом окне, а координаты графического курсора отсчитываются всегда относительно левого верхнего угла окна.

Графический курсор невидим, но его текущие координаты могут быть определены с помощью процедур ***GetX и GetY***.

Синтаксис: ***GetX : integer; GetY : integer***.

Для очистки графического экрана используется процедура без параметров ***ClearDevice***.

После очистки экрана графический курсор будет установлен в верхнем левом углу текущего графического окна.

Процедура без параметров ***ClearViewPort*** очищает текущее графическое окно и заполняет его цветом первой позиции цветовой гаммы.

Задание №1. *Построить на желтом фоне экрана размером 640x350 графическое окно, расположенное симметрично относительно центра экрана с размерами в два раза меньше размеров экрана, и в этом окне построить синюю диагональную прямую и красную точку в левом нижнем углу окна.*

```
Program Gra1;
Uses Graph; {вызов модуля Graph}
Var
Driver, Mode : integer;
Begin
Driver := Detect; {автоматическое распознавание карты}
InitGraph(Driver,Mode,''); {установка графического режима}
SetBkColor(14); {установка желтого цвета фона}
    SetViePort(160,88,479,262,ClipOn);{выделение графического окна
    размером 320x175}
SetColor(1); {выбор синего цвета}
LineRel(319,174); {рисование диагонали}
PutPixel(0,174,4); { отображение красной точки }
Readln;
CloseGraph { возврат в текстовый режим }
end.
```

Предложение ***Readln*** позволяет задержать на экране окно вывода с графическими изображениями до тех пор, пока не будет нажата клавиша <***Enter***> .

Рисование окружности с центром в точке (X, Y) и радиусом ***Radius*** выполняется процедурой ***Circle(X, Y: integer; Radius: word);***

Рисование дуги окружности с центром в точке (***X, Y***), радиусом ***Radius***, углом начала ***StAngle*** и углом конца ***EndAngle***, выполняется процедурой:

Arc(X, Y : integer; StAngle, EndAngle : word; Radius : word).

Углы ***StAngle*** и ***EndAngle*** выражаются в радианах. Рисование происходит против движения часовой стрелки.

Рисование дуги эллипса с центром в точке (***X, Y***), полуосями ***XRadius*** и ***YRadius***, углами начала ***StAngle*** и конца ***EndAngle*** выполняется процедурой:

Ellipse(X, Y: integer; StAngle, EndAngle : word; Radius, YRadius : word).

Углы выражаются в радианах. Рисование осуществляется против часовой стрелки. Для ***StAngle = 0*** и ***EndAngle = 2*** будет нарисован полный эллипс.

Изображение закрашенного сектора круга выполняется процедурой:

PieSlice(X, Y : integer; StAngle, EndAngle : word; Radius : word).

Параметры этой процедуры имеют тот же смысл, что и в процедуре ***Arc***, рассмотренной выше. Шаблон закраски задается процедурой ***SetFillStyle*** (см. ниже).

Например, *требуется построить сектор круга с центром в точке (319, 174), радиусом в 100 точек и представляющим собой левую нижнюю четверть круга*. Для этого достаточно выполнить процедуру ***PieSlice*** со следующими параметрами: ***PieSlice (319,174,180,270,100).***

Процедурой ***Rectangle(X1, Y1: integer; X2, Y2: integer)*** будет построен прямоугольник, противоположные вершины которого имеют координаты (***X1, Y1***) и (***X2, Y2***).

Процедура ***Bar(X1, Y1: integer; X2, Y2: integer)*** в отличие от предыдущей процедуры строит закрашенный прямоугольник.

Допустим, необходимо построить закрашенный в синий цвет прямоугольник с горизонтальной стороной в 200 точек, с вертикальной в 100 точек и с центром в точке (319, 174). Два следующих предложения решают эту задачу:

```
SetColor (1); { выбор синего цвета }
Bar(219,124, 419,224); {вычерчивание закрашенного прямоугольника}
```

Выбор типа линии, которой будет нарисован объект, можно выполнить процедурой:

SetLineStyle(LineStyle: word; Pattern: word; ThickNess:word);

Выбор типа линий выполняется на основании ***LineStyle*** и толщины линии по ***ThickNess***. Если ***LineStyle = UserBitLn***, то тип линии выбирается по системе битов ***Pattern***. Этот параметр рассмотрен при описании процедуры "заливки". Тип линии может быть выражен следующим образом:

SolidLn = 0 - сплошная линия;

DottedLn = 1 - пунктирная линия (из точек);

CenterLn = 2 - осевая линия (из точек и тире);

DashedLn = 3 - штриховая линия;

UserBitLn = 4 - линия, определяемая пользователем.

Толщина линий может быть выражена следующим образом:

NormWidth = 1 - тонкая линия;

ThickWidth = 3 - толстая линия.

Задание №2. *Построить прямоугольник точечной тонкой линией зеленого цвета, противоположные углы которого имеют координаты (10,10) и (319,174).*

```
Program Rect;
Uses Graph;
Var
dr, mode : integer;
Begin
{ инициализация граф. экрана };
dr := detect;
InitGraph (dr, mode,'');
{ Установка тонкой (1) линии точечного типа };
SetLineStyle(DottedLn, 0, 1);
SetColor(2);
{ вычерчивание прямоугольника };
Rectangle(10,10, 319,174);
Readln;
End.
```

Заполнять области стандартными шаблонами позволяет процедура:

SetFillStyle(Pattern : word; Color : word),

где ***Pattern*** задает номер шаблона, а ***Color*** - номер цвета шаблона.

Номера заполняющих шаблонов могут быть выражены следующим образом:

EmptyFill = 0 - заполнение цветом фона;

SolidFill = 1 - сплошное заполнение;

LineFill = 2 - заполнение толстыми горизонтальными линиями;

LtSlashFill = 3 - заполнение наклонными линиями (правый наклон);

SlashFill = 4 - заполнение толстыми наклонными линиями;

BkSlashFill = 5 - заполнение толстыми косыми линиями (левый наклон);

LtBkSlashFill = 6 - заполнение косыми линиями;

HatchFill = 7 - заполнение вертикальной сеткой;

HatchFill = 8 - заполнение наклонной сеткой;

InterleaveFill = 9 - заполнение переплетенными линиями;

WideDotFill = 10 - заполнение точками;

CloseDotFill = 11 - плотное заполнение точками.

Например: *заполнить вертикальной сеткой красного цвета прямоугольник, рассмотренный в предыдущем примере, необходимо использовать следующие предложения:*

```
SetFillStyle(7, 4);
Bar(10,10, 319,174);
```

Заполнение заданным шаблоном области, охватывающей точку с координатами ***(X, Y),*** ограниченной линией, номер цвета которой определен ***Border***, выполняется процедурой ***FloodFill(X, Y: integer; Border: word).***

Шаблон и цвет заполнения области могут быть определены с помощью процедуры ***SetFillStyle*** .

Например, *заполнить наклонными линиями коричневого цвета круг желтого цвета с центром в точке (319, 174) и радиусом 100, необходимо использовать операторы:*

```
SetFillStyle(3, 6);
SetColor(14);
Circle(319, 174, 100);
FloodFill (310, 170, 14);
```

Контроль выполнения графических операций

При выполнении графических процедур возможны следующие ошибочные ситуации:

1.Параметры процедуры не нарушают требований синтаксиса, но подобраны неправильно при этом выполнение процедуры не вызовет никаких изменений.

2.Параметры подобраны правильно, а процедура выполняется неправильно.

3.Определение причин неправильного выполнения графических операций остается за разработчиком программы. Эту задачу упрощает функция ***GraphResult***, позволяющая определить результат завершения графической операции.

Синтаксис: ***GraphResult : integer;***

Если операция закончилась успешно, функция возвращает 0, в противном случае

отрицательное значение, идентифицирующее причину неудачи. Коды ошибок:

grOk -0 -нормальное выполнение графической операции;
grNoInitGraph -1 -графический режим не установлен;
grNotDetected -2 -нет графической карты;
grFileNotFound -3 -файл драйвера устройства не найден;
grInvalidDriver -4 -неподходящий файл драйвера устройства;
grNoloadMem -5 -нет памяти для загрузки драйвера;
grNoScanMem -6 -нет памяти для заполнения области методом сканирования;
grNoFloodMem -7 -нет памяти для заполнения области методом заливки;
grFontNotFound -8 -файл со шрифтами не найден;
grNoFontMem -9 -нет памяти для загрузки шрифта;
grInvalidMode -10 -неправильный графический режим;
grError -11 -другие ошибки;
grIOError -12 -ошибки операции ввода - вывода;
grInvalidFont -13 -ошибочный стиль шрифта;
grInvalidFontNum -14 -ошибочный номер стиля шрифта;

grInvalidDeviceNum -15 -ошибочный номер устройства.

Поскольку двукратный вызов функции ***GraphResult*** (без выполнения между вызовами графической операции) приводит к тому, что вторым результатом всегда будет ***0***, рекомендуется назначить первый результат промежуточной переменной.

Для обработки "неуспешных" графических операций следует создавать процедуры анализа и выхода из соответствующих ситуаций.

Задание № 3. *Построить график функции Y = Sqr(X). При отображении графика функции на экране необходимо выполнить переход от локальной системы координат в систему координат экрана, а также во избежание помех рассчитать граничные значения X, при которых значения Y начинают выходить за пределы экрана.*

```
Program GraficFunction;
Uses Graph;
Var
grDriver : Integer;
grMode : Integer;
ErrCode : Integer;
X, Y, X1, Y1, CX, CY, XG: Integer;
Begin
{*** инициализация графического режима экрана ***}
grDriver := Detect;
109
InitGraph(grDriver, grMode,'');
ErrCode := GraphResult;
If ErrCode=grOk then {инициализация графического экрана прошла
успешно}
begin {*** Построение осей координат ***}
CX := Round(GetMaxX / 2); {для настройки на координаты любого};
CY := Round(GetMaxY / 2); {экрана используем функции GetMaxX и
GetMaxY};
{(CX, CУ) - центр сист. коорд-т};
Line(0, CY, GetMaxX, CY); {вычерчивание оси ординат};
Line(CX, 0, CX, GetMaxY); { - " - абцисс};
{*** Построение графика функции Y = Sqr(X)***};
XG:=Round(Sqrt(20*(GetMaxY-CY)));{определение ширины параболы};
for X:= -XG to XG do
begin
X1 := X + CX; {определение текущей координаты X экрана};
      Y := Sqr(X); {определение ординаты функции};
      Y1 := GetMaxY - Round(( Y / 20 + CY)); {преобразование текущей
ординаты
 функции в текущую координату Y экрана}
```

```
Circle(X1, Y1, 2) { точка, инцидентная параболе, вычерчивается
в виде окружности радиусом в два пикселя}
end;
{*** Вывод на экран пояснительного сообщения ***}
SetTextStyle(0, 0, 2);
OutTextXY(180, 350, 'График функции Y = Sqr(X)');
ReadLn;
CloseGraph;
end
else { возникла ошибка при инициации графического экрана}
WriteLn('Graphics error:',GraphErrorMsg(ErrCode));
 {вывод диагностического сообщения}
End.
```

Графический вывод текста

Для графического режима разработаны специальные процедуры, обеспечивающие вывод сообщений различными шрифтами в горизонтальном или вертикальном направлении с изменением размера символов. Рассмотрим некоторые из них.

Процедура ***Outtext*** выводит текстовую строку ***Txt***, начиная с текущего положения указателя. Заголовок этой процедуры описывается следующим образом:

Procedure OutText (Txt : String);

Процедура ***OutTextXY*** выводит текстовую строку ***Txt***, начиная с заданного координатами X и Y места:

Procedure OutTextXY(X,Y : Integer; Txt : String);.

Процедура ***SetTextstyle*** устанавливает стиль текущего вывода на графический экран. Процедуре передаются в качестве параметров ***Font*** – код шрифта, ***Direct*** - код направления, ***Size*** - код размера шрифта.

Procedure SetTextStyle (Font, Direct, Size ; Word);

Для указания кода шрифта можно использовать определённые в модуле **Graph** константы:

DefaultFont = 0; {Точечный шрифт 8x8}
TriplexFont = 1; {Утроенный шрифт}
SmallFont = 2; {Уменьшенный шрифт}
SansSerifFont = 3; {Прямой шрифт}
GothicFont = 4; {Готический шрифт}

Для задания направления выдачи текста можно использовать константы:

HorizDir = 0; {Слева направо}
VertDir = 1; {Снизу вверх}

Каждый шрифт способен десятикратно изменять свои размеры. Размер

выводимых символов кодируется параметром ***Size***, который поэтому может

иметь значение от 1 до 10. Минимальный размер шрифта, при котором отчётливо различаются все его детали, равен 4.

Процедура с заголовком ***Procedure SetTextJustify (Horiz, vert : Word);***

задаёт выравнивание (***Horiz*** - горизонтальное, ***Vert*** - вертикальное) выводимого текста по отношению к текущему положению указателя или к заданным координатам. выравнивание определяет будет ли текст размещаться левее, правее, выше или ниже указанного места. в качестве значений параметров описываемой процедуры можно использовать следующие константы модуля ***Graph*** const

LeftText = 0; {Указатель слева от текста}

CenterText = 1; {Симметрично слева и справа, вверху и снизу}

RightText = 2; {Указатель справа от текста}

BottomText = 0; {Указатель снизу от текста}

TopText = 2; {Указатель сверху от текста}

Функции ***TextWidth*** и ***TextHeight*** с заголовками соответственно

Function TextWidth (Txt : String): Word;

Function TextHeight(Txt : String): Word;

возвращают длину и ширину в пикселях передаваемой в качестве параметра

строки с учётом текущего стиля вывода.

Для того, чтобы выяснить текущий стиль и выравнивание текста,

используется процедура ***GetTextSettings***

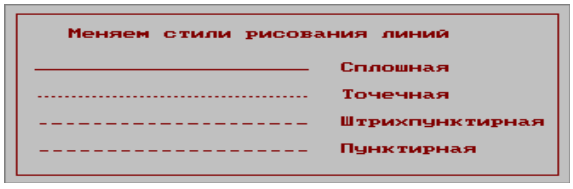
Procedure GetTextSettings (var TextInfo : TextSettingsType);

Тип ***TextSettingsType*** определён в модуле ***Graph*** следующим образом:

```
type
TextSettingsType = record
Font : Word; {Номер шрифта}
Direction : Word; {Направление }
CharSize : Word; {Код размера }
Horiz : Word; {Горизонтальное выравнивание}
Vert : Word; {Вертикальное выравнивание}

end;
```

Задание № 3. *Вывести возможные изображения линий в графическом режиме.*



Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Lines_3;
Uses Graph, Crt; {подключение к программе библиотек Crt и Graph}
Var
Key : Char;  LineStyle : Word; {номер стиля рисования линии}
Style : String; {название стиля } GrError : Integer; {код ошибки графики}
GrDriver, GrMode : Integer; {тип и режим работы графического драйвера}
Begin
GrDriver := Detect; {автоопределение типа графического драйвера}
InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка граф. режима}
GrError := GraphResult;
If GrError<>GrOk then begin Writeln('Обнаружена ошибка!'); Halt; end;
SetBkColor(LightGray); SetColor(Red); {цвет фона и цвет рисования }
OutTextXY(120, 100, 'Рисуем линию от точки (200,200) к точке (400,280)');
Line(200, 200, 400, 280);
Key:=ReadKey; {приостановление исполнения программы}
ClearViewPort; {очистка окна}
OutTextXY(240, 80, 'Рисуем ломанную');
Rectangle(110, 120, 520, 400); {рисование рамки }
MoveTo(Round(GetMaxX/2), Round(GetMaxY/2)); {указатель в центре окна}
Repeat {цикл прерывается нажатием любой клавиши}
LineTo(Random(GetMaxX-250)+120, Random(GetMaxY-210)+120);
Delay(100);
until KeyPressed;
Key := ReadKey; ClearViewPort;
OutTextXY(190, 80, 'Меняем стили рисования линий');
For LineStyle := 0 to 3 do begin
                          SetLineStyle(LineStyle, 0, 1);
                               Case LineStyle of
                               0: Style:='Сплошная';
                               1: Style:='Точечная';
                               2: Style:='Штрихпунктирная';
                               3: Style:='Пунктирная' ; end;
          Line(120, 150+LineStyle*50, 430, 150+LineStyle*50);
          OutTextXY(450, 145+LineStyle*50, Style);   end;
Key:=ReadKey; ClearViewPort; {очистка окна}
OutTextXY(180, 80, 'Меняем толщину рисования линий');
SetLineStyle(0, 0, 1); {толщина 1 пиксел }
Line(140, 200, 430, 200); OutTextXY(450, 195, 'Нормальная');
SetLineStyle(0, 0, 3); {толщина 3 пиксела}
Line(140, 250, 430, 250); OutTextXY(450, 245, 'Тройная');
ReadLn; CloseGraph; {закрытие графического режима}        End.
```

Задание № 4. *Вывести изображения символов в графическом режиме (требует наличия в текущем каталоге файлов шрифтов *.chr).*

Внимание: будет работать только если Turbo Pascal установлен в каталог C:\TP и каталог C:\TP\BGI содержит необходимый файл egavga.bgi.

```
Program Symbols;
Uses Graph, Crt; {подключение к программе библиотек Crt и Graph}
Var
Key : Char;
Font : String; {названия шрифтов }
Size, MyFont : Word;
GrDriver, GrMode : Integer; {тип и режим работы графического драйвера}
Begin
GrDriver := Detect; {автоопределение типа графического драйвера}
InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима }
If GraphResult <> GrOk then Halt;
SetTextStyle(DefaultFont, HorizDir, 2);
OutTextXY(140, 80, 'Меняем размер символов');
OutTextXY(220, 100, 'и цвет фона');
For Size := 0 to 13 do {Size - цвет фона и размер символов}
begin SetBkColor(Size); {изменение цвета фона }
Rectangle(135, 425, 470, 450); {рисование рамки }
SetTextStyle(DefaultFont, HorizDir, 1);
OutTextXY(150, 435, 'Для продолжения нажмите любую клавишу !');
SetTextStyle(DefaultFont, HorizDir, Size);
OutTextXY(250-Size*15, 200, 'HELLO');
Key := ReadKey; ClearViewPort;
end; ReadLn;
SetBkColor(LightGray); SetColor(Red);{цвет фона и цвет рисования }
SetTextStyle(DefaultFont, HorizDir, 2);
{установка шрифта, направления и размера символов}
OutTextXY(70, 100, 'Располагаем строку горизонтально');
SetTextStyle(DefaultFont, VertDir, 2);
OutTextXY(310, 150, 'и вертикально');
Key:=ReadKey; ClearViewPort;
      SetTextStyle(DefaultFont, HorizDir, 2);
{установка шрифта, направления и размера символов}
OutTextXY(220, 30, 'Меняем шрифты');
For MyFont := 0 to 9 do begin {цикл по номерам шрифтов}
Case MyFont of
0: Font:='0 - Точечный (Default)';
1: Font:='1 - Утроенный (Triplex)';
2: Font:='2 - Уменьшенный (Small)';
3: Font:='3 - Прямой (SansSerif)';
```

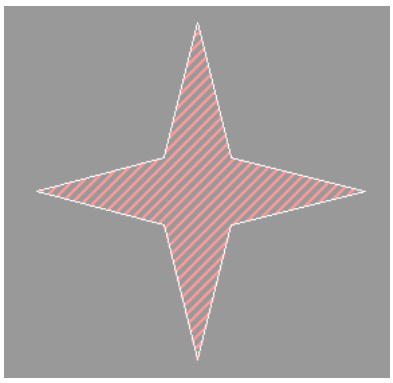
```
4: Font:='4 - Готический (Gothic)';
5: Font:='5 - Рукописный';
6: Font:='6 - Курьер';
7: Font:='7 - Красивый (Таймс Italic)';
8: Font:='8 - Таймс Roman';
9: Font:='9 - Курьер увеличенный';  end;
SetTextStyle(MyFont, HorizDir, 2);
OutTextXY(40, 70+MyFont*35, 'abcdfxyz 0123456789');{вывод текста}
SetTextStyle(DefaultFont, HorizDir, 1);
OutTextXY(410, 80+MyFont*35, Font); end; {вывод названия шрифта}
OutTextXY(380, 60, 'N шрифта Описание'); ReadLn;
CloseGraph; {закрытие графического режима}
End.
```

Задание № 5. *Эта программа рисует закрашенный прямоугольник, меняя случайным образом цвет, тип штриховки и высоту тона звукового сопровождения.*

Внимание: будет работать только если Turbo Pascal установлен в каталог C:\TP и каталог C:\TP\BGI содержит необходимый файл egavga.bgi.

```
Program MusicColor;
Uses Crt, Graph; {подключение к программе библиотек Crt и Graph}
Var
GrDriver, GrMode: Integer; {тип и режим работы графического драйвера}
Begin
GrDriver := Detect; {автоопределение типа графического драйвера}
InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима}
SetColor(White); {установка белого цвета рамки }
Rectangle(130, 130, 460, 370); {рисование рамки }
Randomize; {инициализация датчика случайных чисел}
Repeat {цикл прерывается нажатием любой клавиши}
Sound(Random(2000)); {изменение высоты звука }
Delay(Random(1000)); {задержка }
SetFillStyle(Random(4), Random(16)); {смена типа штриховки и цвета}
Bar(140, 140, 450, 360); {рисование закрашенного прямоугольника}
until KeyPressed;
NoSound; {отмена звука }
CloseGraph; ReadLn; {закрытие графического режима}
End.
```

Задание № 6. *Нарисовать на экране звезду и закрасить её, используя 12 типов штриховки.*



Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Star;
Uses Crt, Graph;
{подключение к программе библиотек Crt и Graph}
Const { массив координат вершин многоугольника (звезды) }
TopsStar: Array[1..18] of Integer = (300, 125, 325, 225, 425, 250,
325, 275, 300, 375, 275, 275, 180, 250, 275, 225, 300, 125);
Var
i, j, GrDriver, GrMode : Integer;
Begin
GrDriver := Detect;
InitGraph(GrDriver, GrMode, 'C:\TP\BGI'); {установка графического режима}
SetTextStyle(DefaultFont, HorizDir, 2); {установка шрифта,
направления и размера символов}
OutTextXY(220, 60, 'S T A R ');
SetTextStyle(DefaultFont, VertDir, 2);
OutTextXY(140, 150, 'S T A R ');
SetTextStyle(DefaultFont, VertDir, 2);
OutTextXY(500, 150, 'S T A R ');
i:=0;
Repeat
j:=i mod 12; { j - остаток от деления i на 12 }
SetFillStyle(j, Random(13)); { штриховка и фон }
FillPoly(9, TopsStar); {рисование и штриховка звезды}
Inc(i); {увеличение i на 1}
        Delay(500)
        until KeyPressed; {завершение цикла нажатием любой клавиши}
CloseGraph
End.
```

Задание № 7. *Показать эффект движения изображения прицела с управлением клавишами клавиатуры и выводом координат центра прицела.*

Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi.**

```
Program Sight;
Uses Crt, Graph; {подключение к программе
библиотек Crt и Graph}
Const Step = 5; {шаг изменения координат центра прицела }
Instr = 'управление движением прицела - стрелки, выход - ESC';
Var
GrDriver, GrMode : Integer; {тип и режим работы графического драйвера}
X, Y : Integer; {координаты центра прицела}
XStr, YStr : String;
Ch : Char;
Procedure MakeSight(X, Y : Integer); {процедура рисования прицела}
Begin SetColor(White);
Circle(X, Y, 80);
SetColor(LightGreen);
Line(X-80, Y, X+80, Y); Line(X, Y-63, X, Y+63); {вывод осей прицела}
SetColor(LightRed); Circle(X, Y, 2); {окружность в центре прицела}
Str(X, XStr); Str(Y, YStr); {перевод координат в строковый тип}
SetColor(Yellow);
OutTextXY(X+5, Y-35, 'x=' + XStr); {вывод координат центра прицела }
OutTextXY(X+5, Y-20, 'y=' + YStr)
End;
Begin
GrDriver := Detect;
InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
SetColor(LightGray);
X := GetMaxX div 2; Y := GetMaxY div 2; {координаты центра экрана}
Rectangle(50, 425, 600, 460); {рисование рамки }
OutTextXY(120, 440, Instr);
MakeSight(X, Y); {рисование прицела в центре экрана}
While TRUE do {цикл работы программы до прерывания по клавише ESC}
begin
Ch := ReadKey;
Case Ch of
#27: begin CloseGraph; Halt(1) end; {выход по клавише ESC}
#75: X: = X-Step; {изменение координат x, y нажатием стрелок}
#77: X: = X+Step; {"влево", "вправо", "вверх", "вниз" }
#72: Y: = Y-Step;
#80: Y: = Y+Step
end;
ClearViewPort; { очистка графического экрана }
SetColor(LightGray); {восстановление рамки с надписью}
```

```
Rectangle(50, 425, 600, 460);
OutTextXY(120, 440, Instr);
MakeSight(X, Y) {рисование прицела в текущих координатах}
end; CloseGraph;    End.
```

Задание № 8. *Вывод изображения объёмных предметов и столбчатых диаграмм.*

Внимание: будет работать только если Turbo Pascal установлен в каталог **C:\TP** и каталог **C:\TP\BGI** содержит необходимый файл **egavga.bgi**.

```
Program Design;
Uses
Graph, Crt; {подключение к программе библиотек Crt и Graph}
Const
Height : Array[1..8] of Integer=(40,150,90,240,190,120,50,90);
{массив высот столбиков диаграммы}
Var
Color : Word; {код цвета}     Key : Char;
i, x, y, y1, h : Integer;
GrDriver, GrMode : Integer; {тип и режим работы графического драйвера}
GrError : Integer; {код ошибки графики}
Begin
GrDriver := Detect; InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
GrError := GraphResult; If GrError <> GrOk then Halt;
y := 120; h := 50; y1 := 140;
SetTextStyle(DefaultFont, HorizDir, 2); {шрифт, направление, размер}
OutTextXY(160, 20, 'Конструируем интерьер');
SetFillStyle(5, LightRed); {тип штриховки и цвет (ярко красный)}
For i := 4 downto 1 do begin {рисование параллелепипедов заданного размера}
Bar3D(75, y1+i*h, 145, y1+(i+1)*h, 60, TopOff); Delay(200);  end;
Bar3D(75 , y1 , 145, y1+h , 60, TopOn); Delay(200);
Bar3D(180, y , 290, y+h , 30, TopOn); Delay(200);
Bar3D(330, 225 , 400, y+4*h , 30, TopOn); Delay(200);
Bar3D(300, y+3*h, 370, y+5*h , 30, TopOn); Delay(200);
Bar3D(370, y+3*h, 440, y+5*h , 30, TopOn); Delay(200);
Bar3D(300, y , 370, y+h , 30, TopOn); Delay(200);
Bar3D(370, y , 440, y+h , 30, TopOn); Delay(200);
Bar3D(442, y , 500, y+5*h , 30, TopOn); Delay(200);
Rectangle(135, 425, 470, 450); {рисование рамки для сообщения}
SetTextStyle(DefaultFont, HorizDir, 1);
OutTextXY(150, 435, 'Для продолжения нажмите любую клавишу !');
Key := ReadKey; ClearViewPort; {очистка окна}
SetTextStyle(DefaultFont, HorizDir, 2);
OutTextXY(100, 20, 'Рисуем столбиковую диаграмму');
```

```
x := 50; Randomize; {инициализация датчика случайных чисел}
For i := 1 to 8 do begin  {цикл по столбикам диаграммы}
Color := Random(12)+1; {задание кода цвета (кроме черного)}
SetFillStyle(i, Color); {задание типа штриховки и цвета}
SetColor(Color);
Bar3D(x, 350-Height[i], x+50, 380, 20, TopOn); {рисование столбика}
x := x+70; {изменение координаты x };
Delay(200) {задержка};   end;
Key := ReadKey; CloseGraph; {Закрытие графического режима}     End.
```

Задание № 9. *Нарисовать прямоугольную систему координат, отобразить в ней заданное множество точек и построить все возможные пары треугольников с вершинами в этом множестве такие, чтобы один треугольник лежал строго внутри другого***.**

Для работы программы необходимо предварительно создать в текущем каталоге текстовый файл dan.dat, содержащий координаты точек множества. Файл должен иметь структуру:

x1 y1 x2 y2 ... xn yn , где 0 < xi < 400, 0 < yi < 600.

Пример файла dan.dat, содержащего координаты десяти точек:

20 20 150 40 90 300 500 400 50 380 110 130 370 290 300 140 70 60 500 170

Пустых строк в файле dan.dat быть не должно.

Внимание: будет работать только если Turbo Pascal установлен в каталог C:\TP и каталог C:\TP\BGI содержит необходимый файл egavga.bgi.

```
Program Triangles; {Составил студент Тезадов С., 1 к. мат. фак. КБГУ}
Uses Crt,Graph;
Const DemoN = 10;
DemoX: array [1..DemoN] of Integer = (20,150,90,500,50,110,370,300,70,500);
DemoY: array [1..DemoN] of Integer = (20,40,300,400,380,130,290,140,60,170);
Var X, Y : Array[1..50] of Integer; {координаты точек множества}
InX, InY : Array[1..50] of Integer; {координаты вершин внутренних}
Flag : Boolean; {треугольников}      Ch : Char;
Coord, Num : String;                 i, j, k, p, i1, j1, k1, n, n1 : Integer;
GrDriver, GrMode, GrError : Integer;
```

Procedure InputOutput; {Описание процедуры считывания координат точек множества из текстового файла dan.dat в массивы X и Y и вывода точек на графический экран }

```
Var f : Text;       a,b : Real;
Begin
Assign(f, 'dan.dat'); {установление связи между физическим }
{файлом dan.dat и файловой переменной f}
{$I-} {- отключаем автоматическую проверку существования файла}
```

```
Reset(f); i:=0; {открытие файла f для чтения}
{$I+}
If IOResult = 0 then begin {если файл существует}
While not eof(f) do {цикл "пока не будет достигнут конца файла"}
begin Read(f,a,b); Inc(i); {считывание из файла f пары координат}
X[i]:=Trunc(a-1); Y[i]:=Trunc(428-b) {преобразование декартовых}
end; {координат в координаты графического экрана}
n:=i; {n - количество введенных точек множества}
Close(f); {закрытие файла f}
Else begin {если файла не существует, то используем множество точек,}
n := DemoN; {заданное в DemoN, DemoX, DemoY.}
For i:=1 to DemoN do begin      x[i] := DemoX[i];
y[i] := 428 - DemoY[i];  end;
end;
SetColor(LightCyan);
OutTextXY(200,30,'Исходное множество точек');
      For i:=1 to n do begin Circle(X[i], Y[i], 2);{рисование и нумерация точек
      множества}
Str(i, Num); OutTextXY(X[i]+4, Y[i]+3, Num;   end;
Ch:=ReadKey; ClearViewPort; {очистка графического окна}
End; {of InputOutput}
      Procedure Drawing_Axes; {описание процедуры рисования осей
      координат}
Begin SetColor(White);
MoveTo(30,0); LineTo(30,430); LineTo(639,430); {оси OX,OY}
OutTextXY(27,0,'^'); OutTextXY(630,427,'>'); {стрелки осей OX, OY}
SetColor(LightGreen);
OutTextXY(18,0,'y'); OutTextXY(630,434,'x');
OutTextXY(25,433,'0');
SetColor(LightMagenta); {установка розового цвета}
For i:=1 to 20 do {нанесение делений и числовых отметок на ось OY}
begin Str(20*(21-i), Coord); j:=i*20+10;
OutTextXY(2, j-5, Coord);
Line(28, j, 30, j);    end;
For i:=1 to 29 do {нанесение делений и числовых отметок на ось OX}
begin Str(20*i,Coord); j:=i*20+30;
If Odd(i) then OutTextXY(j-8, 436,Coord); Line(j,430, j,432);    end;
SetViewPort(31,4,630,429,FALSE) {установка текущего графического окна}
End; {of Drawing_Axes}
Function Inside(i, j, k, p : Integer ) : Boolean;
{функция Inside возвращает TRUE, если точка с номером p
находится внутри треугольника с вершинами в точках i, j, k}
Var S1, S2 : Real;
```

```
Function Area(x1, y1, x2, y2, x3, y3 : Real) : Real;
{функция вычисления площади треугольника}
{с вершинами в точках (x1,y1), (x2,y2), (x3,y3)}
Begin Area:=abs((x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2))/2)
End; {of Area}
Begin S1:=Area(X[i], Y[i], X[j], Y[j], X[k], Y[k]);
{S1 - площадь треугольника с вершинами в точках i, j, k}
S2 := Area(X[i], Y[i], X[j], Y[j], X[p], Y[p]) +
Area(X[j], Y[j], X[k], Y[k], X[p], Y[p]) +
Area(X[k], Y[k], X[i], Y[i], X[p], Y[p]);
{S2 - сумма площадей трех треугольников с вершинами
в точках (i,j,p), (j,k,p), (i,k,p) }
Inside:=S1>S2 - 0.001
End; {of Inside}
Procedure Triangle(x1, y1, x2, y2, x3, y3 : Integer; Color : Byte);
Begin {описание процедуры рисования треугольника цвета Color}
SetColor(Color);
Line(x1, y1, x2, y2);
Line(x2, y2, x3, y3);
Line(x3, y3, x1, y1)
End; {of Triangle}
Begin
GrDriver:=Detect;
InitGraph(GrDriver, GrMode, 'C:\TP\BGI');
GrError:= GraphResult;
If GrError<>GrOk then begin WriteLn(' Ошибка графики!'); Halt end;
Drawing_Axes; {вызов процедуры рисования осей координат}
InputOutput; {вызов процедуры ввода и вывода исходных данных}
Flag:=FALSE;
For i:=1 to n -2 do {циклы по номерам вершин внешнего треугольника}
For j:=i+1 to n -1 do
For k:=j+1 to n do  begin
SetColor(LightCyan); {установка яркоголубого цвета}
For p:=1 to n do {рисование и нумерация точек множества}
begin Circle(X[p], Y[p], 2); {рисование точки}
Str(p, Num);
OutTextXY(X[p]+4, Y[p]+3, Num) {вывод номера точки}  end;
n1:=0; {занесение координат точек, находящихся
внутри треугольника, в массивы InX и InY}
For i1:=1 to n do  begin
If (i1<>i) and (i1<>j) and (i1<>k) and Inside(i,j,k,i1)  then begin Inc(n1);
InX[n1]:=X[i1]; InY[n1]:=Y[i1];   end;    end;
If n1>=3 then {если число точек внутри треугольника не меньше трех,}
```

```
begin Flag:=TRUE; {то строятся внутренние треугольники}
For i1:=1 to n1-2 do {циклы по номерам вершин внутренних}
For j1:=i1+1 to n1-1 do {треугольников}
For k1:=j1+1 to n1 do begin {рисование внешнего треугольника красным цветом}
Triangle(X[i],Y[i],X[j],Y[j],X[k],Y[k],LightRed);
{рисование внутреннего треугольника зеленым цветом}
Triangle(InX[i1],InY[i1],InX[j1],InY[j1],InX[k1],InY[k1], LightGreen);
OutTextXY(80,450,'Найдено решение. Нажмите любую клавишу!');
Ch:=ReadKey;
SetColor(Black); {"стирание" сообщения}
OutTextXY(80,450,'Найдено решение. Нажмите любую клавишу!');
{“стирание” внутреннего треугольника}
Triangle(InX[i1],InY[i1],InX[j1],InY[j1],InX[k1],InY[k1], Black); end;
 end;    {конец циклов по номерам вершин внутренних треугольников}
{"стирание" внешнего треугольника}
Triangle(X[i], Y[i], X[j], Y[j], X[k], Y[k], Black)
end; {конец циклов по номерам вершин внешнего треугольника}
SetColor(White);
If not Flag then OutText('Для данного множества нет решений задачи')
else OutText('РАБОТА ПРОГРАММЫ ЗАВЕРШЕНА');
OutTextXY(80,450,' Нажмите любую клавишу ...');
Ch:=ReadKey;
CloseGraph {закрытие графического режима}
End.
```

Самостоятельная работа № 14.

1. При домашней подготовке составить программы на языке ***Turbo Pascal*** согласно варианту задания.
2. В системе ***Turbo Pascal*** создать файлы программ.
3. Отладить и выполнить программы. Результат вывести на экран.
4. Проанализировать работу операторов и символьных функций.
5. Написать отчет. (Краткий конспект и распечатки программ и ре зультатов.)

Самостоятельная работа состоит из двух задач. Студент выбирает из списка заданий свой индивидуальный вариант и выполняет его.

Вариант № 1

Задание 1.

Построить совокупность *n* концентрических окружностей, окрашенных в случайные цвета.

Задание 2.

Сформировать точку, движущуюся по синусоидальной траектории.

Вариант № 2

Задание 1.

Построить *n*-угольник с заданными координатами вершин (координаты вершин вводятся с клавиатуры).

Задание 2.

Сформировать секундную стрелку часов в движении.

Вариант № 3

Задание 1.

Построить совокупность *n* отрезков, окрашенных в случайные цвета и расположенных случайным образом.

Задание 2.

Сформировать точку, движущуюся по контуру воображаемого квадрата с заданными размерами и расположением.

Вариант № 4

Задание 1.

Построить совокупность попарно связанных *n* точек с заданными координатами (координаты точек задаются с клавиатуры).

Задание 2.

Сформировать точку, движущуюся по контуру воображаемой окружности заданного радиуса и расположения.

Вариант № 5

Задание 1.

Построить совокупность *n* занумерованных горизонтально расположенных квадратов заданного размера (номера внутри квадратов).

Задание 2.

Сформировать движущуюся окружность радиуса *r*, центр которой описывает окружность радиуса *r1*.

Вариант № 6

Задание 1.

Построить совокупность *n* окружностей радиуса *r*, центры которых равномерно распределены по окружности радиуса *r1*.

Задание 2.

Сформировать движущуюся окружность радиуса *r*, центр которой перемещается попеременно в обоих направлениях вдоль горизонтального отрезка с заданными границами.

Вариант № 7

Задание 1.

Построить линейную диаграмму по пяти заданным числам. Закрасить ее части в случайные цвета.

Задание 2.

Сформировать движущуюся окружность радиуса *r*, центр которой перемещается попеременно в обоих направлениях вдоль вертикального отрезка с заданными границами.

Вариант № 8

Задание 1.

Построить совокупность *n* кругов радиуса *r* со случайными координатами их центров, закрашенных в случайные цвета.

Задание 2.

Сформировать точку, движущуюся по синусоидальной траектории.

Вариант № 9

Задание 1.

Построить совокупность *n* кругов радиуса *r*, закрашенных в случайные цвета, центры которых расположены на одной вертикали.

Задание 2.

Сформировать круг, закрашенный разными цветами с обеих сторон, вращающийся вокруг горизонтальной оси, расположенной в середине экрана.

Вариант № 10

Задание 1.

Построить совокупность *n* кругов радиуса *r*, закрашенных в случайные цвета, центры которых расположены на одной вертикали.

Задание 2.

Сформировать точку, движущуюся по контуру воображаемого треугольника с заданными размерами и расположением.

Вариант № 11

Задание 1.

Построить совокупность *n* кругов радиуса *r*, закрашенных в случайные цвета, центры которых расположены на диагонали используемого экрана.

Задание 2.

Сформировать движущийся круг радиуса *r*, центр которого перемещается попеременно в обоих направлениях вдоль вертикального отрезка с заданными границами.

Вариант № 12

Задание 1.

Построить круговую диаграмму по заданным пяти исходным числам. Закрасить ее сектора в случайные цвета.

Задание 2.

Сформировать движущийся прямоугольник заданного размера, центр которого перемещается попеременно в обоих направлениях вдоль наклонного отрезка с заданными границами.

Вариант № 13

Задание 1.

Построить прямоугольник по заданным координатам вершин и закрасить его в случайный цвет.

Задание 2.

Сформировать движущийся прямоугольник заданного размера, центр которого перемещается попеременно в обоих направлениях вдоль

горизонтального отрезка с заданными границами.

Вариант № 14

Задание 1.

Построить столбчатую диаграмму (гистограмму) по заданным пяти числам. Закрасить ее прямоугольники.

Задание 2.

Сформировать движущийся прямоугольник заданного размера, центр которого перемещается попеременно в обоих направлениях вдоль вертикального отрезка с заданными границами.

Вариант № 15

Задание 1.

Изобразить на экране шахматную доску $n_x n$ клеток заданного размера.

Задание 2.

Сформировать квадрат, закрашенный разными цветами с обеих сторон, вращающийся вокруг горизонтальной оси, расположенной в середине экрана.

Вариант №16

Задание 1.

Построить совокупность *n* квадратов с общим центром.

Задание 2.

Сформировать квадрат, закрашенный разными цветами с обеих сторон, вращающийся вокруг вертикальной оси, расположенной в середине экрана.

Вариант № 17

Задание 1.

Построить окружность, состоящую из заданного числа дуг случайных цветов.

Задание 2.

Сформировать движущуюся окружность радиуса *r*, центр которой перемещается попеременно в обоих направлениях вдоль наклонного отрезка с заданными границами.

Лабораторная работа №12. *Работа с использованием подпрограмм.*

Цель работы:

1. Изучение приемов программирования с ипользованием подпрограмм (функций, процедур).
2. Приобретение практических навыков в работе с подпрограммами.

Подпрограммы

Подпрограммы представляют собой относительно самостоятельные фрагменты программы, оформленные особым образом и снабженные собственным именем. Упоминание этого имени в тексте основной программы или другой подпрограммы называется ***вызовом подпрограммы***.

Подпрограммы представляют собой инструмент программирования, с помощью которого любая программа (или подпрограмма) может быть разбита на ряд относительно независимых друг от друга частей. Такое разбиение может оказаться целесообразным по следующим причинам:

1. Средство экономии памяти. Каждая подпрограмма существует в единственном экземпляре и может быть предназначена для выполнения вычислительного процесса, повторяющегося в различных точках основной программы или другой подпрограммы. Обращаться к подпрограмме можно многократно из соответствующих мест вызывающей программы. При вызове подпрограммы управление передается последовательности операторов, образующих тело подпрограммы, а с помощью передаваемых подпрограмме параметров нужным образом модифицируется реализуемый в ней алгоритм.
2. Использование подпрограмм позволяет применить методику так называемого ***нисходящего проектирования***. При этой методике алгоритм решения задачи представляется в виде последовательности относительно крупных подпрограмм, реализующих самостоятельные части алгоритма. Подпрограммы в свою очередь могут разбиваться на менее крупные подпрограммы нижнего уровня и т.д. Последовательное структурирование может продолжаться до тех пор, пока реализуемые алгоритмы не станут простыми для программирования.

Текст подпрограммы помещается в разделе описания основной программы и состоит из заголовка, раздела описаний и раздела исполняемых операторов.

Разделы описаний и исполняемых операторов представляют собой блок. Все описания объектов (констант, типов, переменных, подпрограмм и т.д.) считаются локальными по отношению к тому блоку, в котором они охарактеризованы. Это означает, что соответствующие охарактеризованным объектам имена могут употребляться в принятом смысле (иначе говоря, считаются видимыми) только в той части текста, которая относится к данному блоку. Такой фрагмент текста называется ***областью действия этих имен***.

Если подпрограмма является составной частью другой подпрограммы, то текст внутренней подпрограммы помещается в описательной части внешней подпрограммы. Таким образом, возникает разделение объектов на ***глобальные*** и ***локальные***. Глобальными считаются такие объекты, которые оказываются видимыми из любого места программы (в т.ч. и из внутренних подпрограмм). А локальные объекты видимы лишь внутри того блока, где они охарактеризованы. Кроме объектов, охарактеризованных в описательной части подпрограммы, в них могут быть использованы и объекты, характеристики которых определены в заголовке подпрограммы. Они называются ***формальными параметрами*** и используются при обмене данными между подпрограммой и основной программой.

Оформление внутренних блоков для подпрограмм, функций и процедур идентично *и имеет следующий вид:*

Function *<имя функции>(<список формальных параметров>):<имя типа>;*

где <***function***> – служебное слово*;*

<имя функции> – идентификатор выбираемый пользователем в соответствии с общими правилами языка программирования;

<список формальных параметров> – аргументы функции с указанием их типа, используемые для обмена информацией между функцией и точкой вызова; *<имя типа>* – тип значения, вырабатываемого функцией.

Список формальных параметров состоит из отдельных частей, отделяемых друг от друга точкой с запятой «;». Каждая часть включает в себя список параметров одного и того же типа и имеет вид*:*

<список имен переменных> : <имя типа>;

Где *<список имен переменных>* – имена формальных параметров, разделенных запятыми;

<имя типа> **–** общий тип этих параметров.

Задание № 1. Написать подпрограмму в виде функции, предназначенную для вычисления целой степени какого -либо вещественного числа или выражения вещественного типа.

```
Function Power (Num : real; Expon : integer) : real;
{Эта функция возводит вещественное число Num в целочисленную степень Expon}
Var
Count : integer;
Res : real;
Begin
If Expon = 0 Then Power:=1  Else  begin
                    Res:=Num;
                          For Count:=2 to Abs(Expon) do Res:=Res*Num;
                    If Expon < 0 Then Power:=1/Res Else Power:=Res;  End;
End.
```

Внимание. В разделе исполняемых операторов подпрограммы (функции) всегда должен присутствовать хотя бы один оператор присваивания, в левой части которого должна быть указана переменная, имя которой совпадает с именем функции. Через этот оператор вычисленное значение передается в точку вызова.

Для того чтобы воспользоваться подпрограммой (функцией), к ней следует обратиться, задав ее аргументам конкретные значения.

Обращение к функции имеет вид :

<Имя функции>(<список фактических параметров>);

где *<фактические параметры>* – это объекты, указанные в обращении к подпрограмме. В списке они отделяются друг от друга запятыми «,». В

качестве фактических параметров могут выступать константы, переменные, арифметические выражения. Соответствие между формальными и фактическими параметрами устанавливается по порядку их следования. Следовательно, списки параметров в имени подпрограммы и обращении к ней должны быть согасованными по числу параметров, порядку их следования и типу принимаемых значений. Обращение к функции ***Power*** *может быть оформлено в виде:*

Hypotenuse:=SgRt(Power(a,2)+Power(b,2));

Описание подпрограммы (процедуры) включает заголовок, раздел описаний и раздел исполняемых операторов:

Procedure *<имя процедуры>(<список формальных параметров>);*
<раздел описаний>
Begin
<раздел операторов>
End.

Внимание. В отличие от заголовка функции в заголовке процедуры не указывается тип получаемого результата. Эта разница возникает из-за того, что результатом работы процедуры, в отличие от функции, может быть не только единственное значение, но и множество значений. Результат работы процедуры может иметь и невычислительный характер (например, организация вывода или ввода данных).

Формальные параметры, указываемые в заголовке процедуры, могут быть разделены на две разновидности: параметры - переменные и параметры - значения. Если какая-либо группа переменных должна рассматриваться как параметры-переменные, то перед этой группой ставится служебное слово ***var***. *Например:*

Procedure *MyProcedure (****var*** *a,b : real; c : word);*

Определение формального параметра тем или иным способом существенно для вызывающей программы. Если формальный параметр объявлен как параметр-переменная, то при вызове подпрограммы ему должен соответствовать фактический параметр в виде переменной нужного типа.

Если формальный параметр объявлен как параметр-значение, то при вызове ему может соответствовать либо переменная, либо произвольное выражение соответствующего типа. Замена формальных параметров на фактические при обращении к процедуре осуществляется следующим образом.

Если параметр определен как параметр-значение, то перед вызовом подпрограммы вычисляется значение фактического параметра и полученный результат передается подпрограмме. Любые возможные изменения в подпрограмме параметра–значения никак не воспринимаются вызывающей программой, так как в этом случае изменяется только копия фактического параметра.

Если параметр определен как параметр-переменная, то при вызове подпрограммы передается сама переменная, а не ее копия (фактически в подпрограмму передается адрес переменной). Изменение параметра–переменной приводит к изменению самого фактического параметра в вызывающей программе.

Таким образом, параметры-переменные можно рассматривать как выходные параметры, с помощью которых результат, полученный в подпрограмме, передается в вызывающую программу. Параметры-значения могут использоваться как входные параметры, с их помощью в подпрограмму может передаваться информация, необходимая для работы подпрограммы.

Следует отдельно отметить особенность обмена данными между подпрограммой и вызывающей программой в случае, когда в качестве таких данных используются массивы. Эта особенность связана с тем обстоятельством, что типом любого параметра в списке формальных параметров может быть только стандартный тип языка или ранее объявленный.

Ошибочным, например, будет следующее объявление процедуры:

***Procedure** S(a : **array** [1..10] **of** real);*

Здесь в списке формальных параметров используется ранее не объявленный нестандартный тип - диапазон (1..10). В связи с отмеченной особенностью в тех случаях, когда в качестве данных, используемых при обмене между процедурой и вызывающей программой, предполагается использовать массивы, их тип следует объявить во внешней по отношению к процедуре программе. Например:

***Type** Atype = **array** [1..10] **of** real;*
***Procedure** S(a : Atype);*

Следует также иметь в виду, что реальный размер массива может быть меньше, чем объявленный при его описании. Поэтому при обмене исходными данными между процедурой и вызывающей программой вместе с самим массивом следует передать и его реальные размеры.

Задание № 2. Составить процедуры, в которых по трем векторам (одномерным массивам): ***x** = [x1, x2, ..., xn]; **y** = [y1, y2, ..., yl]; **z** = [z1, z2, ..., zk], где **n, l, k** – длина соответствующего вектора строится матрица **A**, строками которой являются элементы векторов **x, y, z**, отсортированные предварительно в порядке возрастания значений их элементов. Длина строк в **A** равна минимальной длине одного из трех исходных векторов.*

```
{Вариант 1 решения задачи (без включения подпрограммы):}
Var i,j,n,l,k,m : byte;
MinX,MinY,MinZ : real;
x,y,z,xx,yy,zz : array [1..100] of real;
Matr : array [1..100,1..3] of real;
Begin
Write('Ввести n, l, k '); ReadLn(n, l, k);
WriteLn('Ввести последовательно массивы x, y, z');
```

```
For i:=1 to n-1 do    Read(x[i]);
ReadLn(x[n]);
For i:=1 to l-1 do   Read(y[i]);
ReadLn(y[l]);
For i:=1 to k-1 do   Read(z[i]);
ReadLn(z[k]);
{Организация сортировки исходных массивов}
i:=1;
Repeat
MinX:=x[i];
For j:=i+1 to n do  If x[j] < MinX  Then begin
                                        MinX:=x[j];
                                              x[j]:=x[i];
                                        x[i]:=MinX;  End;
i:=i+1;
Until i > n-1;
i:=1;
Repeat
MinY:=y[i];
For j:=i+1 to l do  If y[j] < MinY  Then begin
                                        MinY:=y[j];
                                        y[j]:=y[i];
                                        y[i]:=MinY;  End;
i:=i+1;
Until i > l-1;
i:=1;
Repeat
MinZ:=z[i];
For j:=i+1 to k do  If z[j] < MinZ  Then begin
                                        MinZ:=z[j];
                                        z[j]:=z[i];
                                        z[i]:=MinZ;   End;
i:=i+1;
Until i > k-1;
{Выбор длины строки в матрице A}
If (n < l) and (n < k)  Then m:=n  Else if (l < n) and (l < k)  Then m:=l
                                        Else m:=k;
                                        {Формирование матрицы A}
                                  For i:=1 to 3 do
                                  For j:=1 to m do  If i=1 Then Matr[i,j]:=x[j]
                                        Else if i=2  Then Matr[i,j]:=y[j]
                                              Else Matr[i,j]:=z[j];
                                                    {Вывод матрицы A}
```

```
                WriteLn('Матрица результата работы программы');
                        For i:=1 to 3 do   Begin
                        For j:=1 to m-1 do Write(Matr[i,j]:5:2, ' ');
                        WriteLn(Matr[i,m]:5:2);   End;
{Конец программы}
Write('Нажми Enter'); ReadLn;  END.
{Вариант 2 решения задачи (с подпрограммой-процедурой)}
Type a = array [1..100] of real;
b = array [1..100,1..3] of real;
Var i,j,n,l,k,m : byte;
x,y,z,xx,yy,zz : a;
Matr : b;
{-----------------------------------------}
Procedure Ord(w : a; nn : byte; var ww : a);
Var MinW : real;
Begin {Тело процедуры}
{Организация сортировки исходных массивов}
i:=1; MinW:=w[1];
Repeat
For j:=i+1 to nn do
If w[j] < MinW
Then MinW:=w[j];
ww[i]:=MinW; i:=i+1;
Until i > nn;
End;
{-----------------------------------------}
BEGIN
{Начало основной программы}
Write('Ввести n, l, k '); ReadLn(n, l, k);
WriteLn('Ввести последовательно массивы x, y, z');
For i:=1 to n-1 do  Read(x[i]);
ReadLn(x[n]);
For i:=1 to l-1 do  Read(y[i]);
ReadLn(y[l]);
For i:=1 to k-1 do  Read(z[i]);
ReadLn(z[k]);
{Обращение к процедуре Ord для сортировке исходных массивов}
Ord(x,n,xx);
Ord(y,l,yy);
Ord(z,k,zz);
{Выбор длины строки в матрице A}
If (n < l) and (n < k)  Then m:=n  Else if (l < n) and (l < k) Then m:=l
                                        Else m:=k;
```

```
                    {Формирование матрицы A}
                    For i:=1 to 3 do
                    For j:=1 to m do  If i = 1 Then Matr[i,j]:=x[j]
                                      Else if i = 2 Then Matr[i,j]:=y[j]
                                           Else Matr[i,j]:=z[j];
                                           {Вывод матрицы A}
WriteLn('Матрица результата работы программы');
For i:=1 to 3 do
For j:=1 to m-1 do  Write(Matr[i,j]:5:2, ' ');
WriteLn(Matr[i, m]);
{Конец программы}
WriteLn('Нажми Enter'); ReadLn;  END.
```

Для иллюстрации рассмотренных выше понятий далее приводится программа линейной интерполяции. Содержание задачи, реализованной в программе, заключается вследующем. Исходными данными являются значения функции **y** *при некоторых значениях аргумента* **x**. *При этом известно, что* **y** *изменяется в зависимости от* **x** *по линейному закону. Требуется вычислить значения функции при заданных значениях аргумента* **x**, *называемых узлами интерполяции.*

```
     Uses CRT; {стандартный модуль для очистки экрана}
Const l = 20; {определение максимального размера массива}
Type mas = array [1..l] of real; {определение структуры массивов
путем объявления типа}
{-----------------------------------------}
Procedure Intln(x,y : mas; n : byte; xa : real; var ya : real);
{заголовок процедуры: x, y - исходные массивы; n - их рабочая длина;
xa - значение аргумента, при котором определяется интерполируемое
значение функции; значения x, y, n, xa будут определены приобращении к
процедуре; ya - искомое значение функции, оно будет передано в основную
программу}
Var i,i1 : byte;
Begin {Начало тела процедуры}
For i:=2 to n do  Begin
If (x[i]-xa) >= 0  Then begin
CRT (cathode-raytube) [сиарти] – катодно –лучевая трубка .
i1:=i-1;
ya:=y[i1]+(y[i1+1]-y[i1])*(xa-x[i1])/(x[i1+1]-x[i1]); End;
End;
End;
{-----------------------------------------}
{Основная (вызывающая) программа}
Var xx,yy,xint,yint : mas;
     lxy,lxint,i : byte;
```

```
BEGIN {Начало основной программы}
ClrScr;
Write('lxy, lxint = '); ReadLn(lxy, lxint);

{Ввод рабочего размера аргумента (x) и функции(y)}
WriteLn('x, y = '); {Ввод значений аргумента и функции}
For i:=1 to lxy do  Read(xx[i], yy[i]);
WriteLn;
Write('xint = '); {Ввод значений аргумента, при которых находятся}
      For i:=1 to lxint do  Read(xint[i]); {Интерполированные значения
      функции}
WriteLn;
For i:=1 to lxint do
IntLn(xx,yy,lxy,xint[i],yint[i]); {Обращение к процедуре IntLn}
{Переменные, указанные в скобках – это фактические параметры;
      значения xx, yy, lxy, xint передаются в процедуру; значение yint
      получается из процедуры}
WriteLn('xint, yint = '); {Вывод заголовка результатов}
For i:=1 to lxint do
Write(i,' ',xint[i]:5:2,' ',yint[i]:5:2,'**');{Вывод xint, yint}
ReadLn;  END.
```

Методы сортировки данных

Сортировка – это упорядочение элементов массива в определенном порядке:

1. По возрастанию – от меньшего к большему;
2. По убыванию – от большего к меньшему;
3. Символьные строки – по алфавиту.

При решении задачи сортировки обычно выдвигается требование минимального использования дополнительной памяти, из которой вытекает недопустимость применения дополнительных массивов.

Для оценки быстродействия алгоритмов различных методов сортировки, как правило, используют два показателя:

1. Количество присваиваний;
2. Количество сравнений.

Все методы сортировки можно разделить на две большие группы:

1. Прямые методы сортировки;
2. Улучшенные методы сортировки.

Прямые методы сортировки по принципу, лежащему в основе метода, в свою очередь разделяются на три группы:

1. Сортировка вставкой (включением);
2. Сортировка выбором (выделением);
3. Сортировка обменом (‘пузырьковая’ сортировка).

Улучшенные методы сортировки основываются на тех же принципах, что и прямые, но используют некоторые «оригинальные идеи» для ускорения процесса сортировки.

Прямые методы на практике используются довольно редко, так как имеют относительно низкое быстродействие. Однако они хорошо показывают суть основанных на них улучшенных методов. Кроме того, в некоторых случаях (как правило, при небольшой длине массива или особом расположении элементов массива) некоторые из прямых методов могут даже превзойти улучшенные методы.

Сортировка вставкой

Принцип метода строится на разделении массива на две части: *отсортированную* и *не отсортированную*. Элементы из не отсортированной части поочерёдно выбираются и вставляются в отсортированную часть так, чтобы не нарушить в ней упорядоченность элементов.

В начале работы алгоритма в качестве отсортированной части массива принимают только один первый элемент, а в качестве не отсортированной части – все остальные элементы.

Таким образом, исполнение алгоритма состоит из *(n – 1) -го* прохода (n-размерность массива), каждый из которых будет включать четыре действия:

1. Взятие очередного *i - го* не отсортированного элемента и сохранение его в дополнительной переменной.
2. Поиск позиции *j – ого* элемента в отсортированной части массива не нарушит упорядоченности элементов.
3. Осуществление сдвига элементов массива от (*i- 1)- го* до (*j- 1)- го* вправо, чтобы освободить найденную позицию для вставки элемента.
4. Вставка взятого элемента в найденную j- ю позицию.

Для реализации данного метода можно предложить несколько алгоритмов, которые могут отличаться способом поиска позиции вставки.

Программа, реализующая рассмотренный алгоритм, будет иметь следующий вид:

```
Program InsertionSort;
Uses Crt;
Const
N= 20; {длина массива}
Type
Tvector= array [1..n] of real;
Va
Vector : Tvector;
B : Real;
I, j, k : integer;
Begin
ClrScr;
```

```
Writeln ('Введите элементы массива:');
For i:=1 to n do Read (Vector[i]); Readln;
{-----------------------------------------------------------}
for i:= 2 to n do
begin
B:= Vector[i]; {Взятие не отсортированного элемента
{Цикл поиска позиции всиавки}
j:= 1;
while (B> Vector[j]) do
j:= j+1; {После окончания цикла индекс j фиксирует позиции вставки}
{Цикл сдвига элементов для освобождения позиции вставки}
for k:= i-1 downto j do
Vector[k+1] := Vector [k];
{Вставка взятого элемента на найденную позицию }
Vector[j] := B;
End;
{-------------------------------------------------------------------}
writeln('Отсортированный массив:');
for i := 1 to n do write (Vector[i]:8:2);
writeln;
end.
```

Сортировка выбором

Принцип метода основан на нахождении (выборке) в массиве элемента с минимальным значением на интервале от 1- го элемента до n- го (последнего) элемента и перемещение его на место первого элемента. Затем находим элемент с минимальным значением на интервале от 2- го до n –го элемента и меняем местами со вторым элементом. И так далее для всех элементов до n-1-го.

Программа, реализующая метод выбора, будет иметь следующий вид:

```
Program SelectionSort;
Uses Crt;
Const
N= 20 ; {длина массива}
Type
Tvector = array [1..n] of real;
Var
Vector : Tvector;
Min : Real;
Imin, S : Integer;
```

```
I: Integer;
Begin
ClrScr;
Writeln ('Введите элементы массива:');
For i:= 1 to n do Read (Vector[i]); Readln;
{-----------------------------------------------------------------}
for S:=1 to n-1 do
begin
{Поиск минимального элемента в диапазоне}
{от S- го элемента до n- го}
min := vector[S];
Imin := S;
For i := S+1 to n do
If vector[i] < min then
Begin
Min:= vector[i];
Imin := i;
End;
        {Обмен местами минимального и S –го элементов}
        vector[imin] := vector[S];
vector[S] := Min;
end;
{-----------------------------------------------------------------}
writeln('Отсортированный массив:');
for i:= 1 to n do write (vector[i] :8:2);
writeln;
end.
```

Сортировка обменом ('пузырьковая' сортировка)

Принцип метода состоит в сравнении 2-х соседних элементов:

Если они не соответствует заданному условию упорядоченности, то их меняют местами. Далее берутся два следующих соседних элемента и так далее до конца массива.

После одного такого прохода на последней n- ой позиции массива будет стоять максимальный элемент.

Поскольку максимальный элемент уже стоит на своей последней позиции, то второй проход обменов выполняется до n-1- го элемента. И так далее. Всего требуется n-1 проход.

Программа, реализующая метод обмена ('пузырька'), будет иметь следующий вид:

```
Program BubleSort;
Uses Crt;
Const
N= 20; {Длина массива}
Type
Tvector = array [1..n] of real;
Var
Vector : Tvector ;
B : Real;
I, k: integer;
Begin
ClrScr;
Writeln {‘Введите элемент массива:’};
For i:= 1 to n do Read (Vector[i]); Readln;
{-------------------------------------------------------}
for k:= n downto 2 do
{‘Всплывание ’ очередного максимального элемента на k- ю позицию }
for i:= 1 to k-1 do
if Vector[i] > Vector[i+1] then
begin
B:= Vector[i];
Vector[i+1] :=B
End;
{-----------------------------------------------------------------}
writeln {‘Отсортированный массив:’};
for i:= 1 to n do write (Vector[i]:8 :2);
writeln;
end.
```

Самостоятельная работа № 15

1. Набрать, отладить и выполнить программы, реализующие алгоритмы вашего индивидуального задания.
2. Во всех заданиях полученный массив отсортировать. Блок сортировки оформить ввиде подпрограммы (функция, процедура).
3. Составить блок-схему.
4. Проанализировать работу операторов, пользуясь отладочными режимами.
5. Составить краткий конспект. Защитить работу.

Самостоятельная работа состоит из двух задач. Студент выбирает из списка заданий свой индивидуальный вариант и выполняет его.

Вариант №1

Задание 1.

Определить значение и номер элемента массива *X[n]*, наиболее близкого к среднему арифметическому всех элементов массива. Вывести массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Все элементы строки умножить на среднеарифметическое значение данной строки и полученные значения поместить в массив *Y[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант №2

Задание 1.

Дан массив целых чисел *X[n]*. Найти наибольший элемент в массиве, начиная с этого элемента все остальные элементы умножить на наибольший, а полученный результат возвести в квадрат. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Каждое значение элемента матрицы возвести в квадрат, к полученному результату прибавить его адрес (n+m). Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант №3

Задание 1.

Дан массив целых чисел *X[n]*. Определить количество и сумму положительных элементов массива. Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Вычесть поэлементно из каждого столбца, кроме *i*-го, *i*-й столбец. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант №4

Задание 1.

Определить количество и произведение четных элементов массива *X(n)*. Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Вычесть поэлементно из каждой строки, кроме *i*-й, *i*-ую строку. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант №5

Задание 1.

Определить количество элементов массива *X[n]*, заканчивающихся цифрами 5 или 7. Найти сумму элементов, заканчивающихся цифрами 5, и разность элементов, заканчивающихся цифрами 7. Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Даны матрицы целых чисел *X[n, m]* и *Y[n, m]*. Сформировать матрицу и поместить ее в *X[n, m]* по следующему правилу *X[n,n]= X[n,n]*Y[n,n]*. Вывести исходные и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 6

Задание 1.

Определить для массива *X[n]* алгебраические суммы четных и нечетных элементов массива. Вывести исходный массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Определить максимальный и минимальный элементы матрицы и поменять их местами. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 7

Задание 1.

Определить сумму отрицательных и количество положительных элементов массива *X[n]*. Вывести массив и полученные значения. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Найти наибольший и наименьший элементы массива, начиная со следующего за наименьшим, все элементы увеличить на наибольшее значение элемента, а сумму умножить на наименьшее значение. Вывести полученную матрицу и адреса найденных значений. Значения буквенных переменных задавать с клавиатуры.

Вариант № 8

Задание 1.

Определить сумму элементов массива *X[n]* с нечетными номерами. Вывести массив и полученную сумму. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новую матрицу, элементы строк которой больше элементов исходной матрицы на величину

максимального элемента соответствующей строки исходной матрицы. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 9

Задание 1.

Сформировать массив *Y* из элементов массива *X[n]*, изменив значения всех четных элементов в соответствии с формулой $(X[n]+n)^2$. Вывести оба массива. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Найти наибольший элемент массива, начиная со следующего, все элементы массива увеличить на наибольший элемент, а все первые элементы массива уменьшить на наибольший элемент. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 10

Задание 1.

Определить сумму элементов массива *X[n]* с четными номерами. Вывести массив и полученную сумму. Исходный массив элементов получить с помощью датчика случайных чисел. Возвести в квадрат все элементы исходного массива *X[n]*. Вывести полученный массив. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новый массив *Y[n]*, состоящий из среднеарифметических значений элементов строки исходной матрицы *X[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 11

Задание 1.

Сформировать массив *Y* из элементов массива *X[n]*, расположив сначала положительные, а потом отрицательные элементы. Вывести оба массива. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новый массив *Y[n]*, состоящий из суммы квадратов элементов строки исходной матрицы *X[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 12

Задание 1.

Исходный массив элементов получить с помощью датчика случайных чисел. Определить сумму и количество положительных элементов массива *X[n]*. Вывести исходный массив и полученные значения. Количество элементов

задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Сформировать новую матрицу, элемент которой получен возведением в квадрат соответствующего элемента *X[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 13

Задание 1.

Определить среднее арифметическое значение элементов массива *X[n]*. Вывести исходный массив и полученное значение. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Исходную матрицу получить с помощью датчика случайных чисел. Элементы последней строки прибавить ко всем элементам других строк. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Вариант № 14

Задание 1.

Определить наибольший элемент массива *X[n]*. Вывести исходный массив, этот элемент и его порядковый номер. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Определить номер строки матрицы, сумма элементов которой минимальна. Вывести исходную матрицу, минимальную сумму и номер соответствующей строки. Значения буквенных переменных задавать с клавиатуры.

Вариант № 15

Задание 1.

Переписать элементы массива *X[n]* в массив *Y[n]* в обратном порядке. Вывести оба массива. Определить наименьший элемент массива *Y[n]* и его месторасположение. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Определить номер строки матрицы, сумма элементов которой максимальна. Вывести исходную матрицу, максимальную сумму и номер соответствующей строки. Значения буквенных переменных задавать с клавиатуры.

Вариант № 16

Задание 1.

Определить наименьший элемент массива *X[n]*. Вывести исходный массив, этот элемент и его порядковый номер. Исходный массив элементов

получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новый массив *Y[n]*, состоящий из суммы строк исходной матрицы *X[n, m]*. Вывести исходную матрицу и полученный массив. Значения буквенных переменных задавать с клавиатуры.

Вариант № 17

Задание 1.

Переписать все нечетные элементы исходного массива *X[n]* в массив *Y[n]*. Определить сумму элементов массива с нечетными номерами. Вывести полученную сумму. Исходный массив элементов получить с помощью датчика случайных чисел. Значения буквенных переменных задавать с клавиатуры.

Задание 2.

Дана матрица целых чисел *X[n, m]*. Получить новый массив *Y[n]*, состоящий из минимальных элементов строки исходной матрицы *X[n, m]*. Вывести исходную и полученную матрицы. Значения буквенных переменных задавать с клавиатуры.

Лабораторная работа №13. *Работа с использованием файлов.*

Цель работы:

1. Изучение приемов программирования с использованием файловых структур (ввод и вывод данных).
2. Приобретение практических навыков в работе с файлами.

При обработке больших объемов данных более эффективным способом обмена яляется использование файлов, заранее подготовленных с помощью каких-либо текстовых редакторов. Результаты расчетов также можно поместить в файл, который в дальнейшем может обрабатываться с помощью программных средств.

Для того чтобы программа, написанная на языке ***Turbo Pascal***, могла работать с файлом данных, в ней должна быть предусмотрена специальная файловая переменная. При использовании наиболее часто встречающихся типов данных (целых, вещественных, символьных и т.д.) такая переменная в описательной части программы характеризуется типом ***“text”.***

Например: **Var** <имя переменной> : text;

Переменная типа ***“text”*** предназначена для связи программы с одним текстовым файлом. Он представляет собой последовательность символов, разбитую на строки переменной длины. Для работы с файловыми переменными необходимо установить соответствие между рассматриваемой в данный момент переменной и физическим файлом – именованной областью памяти на каком-либо внешнем носителе (как правило, на диске). Файловая переменная связывается с именем файла с помощью стандартной процедуры ***Assign***:

Assign(<файловая переменная>, <имя файла>);

где *<файловая переменная>* – правильный идентификатор, объявленный в описательной части программы как переменная файлового типа;

<имя файла> – текстовое выражение, содержащее имя файла или путь к нему.

Пример: **Var** Data, Result : text;

Assign(Data, 'D:\1CURS\Data.dat');

Assign(Result,'D:\1CURS\Result.dat');

Перед обращением к файлу необходимо его открыть. Для этого предусмотрены две следующие процедуры:

Reset(*<имя файловой переменной>*); открывает существующий файл для чтения из него информации;

ReWrite(*<имя файловой переменной>*); открывает существующий файл для записи в него информации.

К моменту обращения к подпрограмме ***Reset*** соответствующий файл уже должен существовать на диске. Подпрограмма ***Rewrite*** создает новый файл на диске, если к моменту его выполнения файл с указанным именем не был создан ранее.

Внимание! При использовании процедуры записи следует иметь в виду, что в тех случаях, когда открываемый для записи файл уже существует, все данные, хранившиеся в нем до обращения, будут стерты.

Чтение и запись данных в файл становятся возможными только после его открытия. Чтение информации из файла производится с помощью операторов:

Read(*<имя файловой переменной>, <список ввода>*);

ReadLn(*<имя файловой переменной>, <список ввода>*);

Для записи в файл может использоваться один из следующих операторов:

Write(*<имя файловой переменной>, <список ввода>);*

WriteLn(*<имя файловой переменной>, <список ввода>*);

После завершения работы с файлами они должны быть закрыты с помощью оператора:

Close (*<имя файловой переменной>*);

Задание № 1. *Составить программу формирования файла F, состоящего из целых чисел. Программа показывает работу с файловой переменной F и внешним файлом DAT.TXT. В файл заносятся N= 6 записей, каждая из которых представляет собой целое число.*

```
Program  ZAP_TIP; {Имя программы}
Var {Раздел описания переменных}
F:file of integer;
x,i:Integer;
Begin
Assign(F,'DAT.TXT');{подключение файла 'DAT.TXT' к файловой
переменной F}
Rewrite(F); {открытие файла на запись}
```

```
Writeln(' Введите 6 целых чисел '); {Вывод сообщения}
For i:=1 to 6 do  Begin {цикл для ввода данных}
Read(x); {чтение с клавиатуры}
Write(F,x); {запись на магнитный диск в файл}
End; {конец цикла}
Close(F); {закрытие файла}
END. {Конец программы}
```

Задание № 2. *Составить программу чтения файла F, состоящего из целых чисел. Программа выполняет работу с файловой переменной F и внешним файлом DAT.TXT. Из файла выводятся целые числа.*

```
Program CTEN_TIP; {Имя программы}
Var {Раздел описания переменных}
F:file of integer;
x,i:integer;
Begin {начало операторной части программы}
writeln('Чтение типизированного файла'); {Вывод сообщения}
Assign(F,'DAT.TXT'); {подключение файла 'DAT.TXT' к файловой
переменной F}
Reset(F); {открытие файла для чтения}
While not eof (F) DO {цикл для чтения данных с магнитного диска на
экран}
Begin
Read(F,x); {чтение данных из файла в переменную}
Write(x,' '); {вывод данных из переменной на экран}
End;
Close(F); {закрытие файла}
End. {Конец программы}
```

Задание № 3. *Составить программу ввода текстового файла с именем 'TEX.TXT', представляющий собой список необходимой техники. Сделать так, чтобы в каждой строке файла записывалось одно наименование. На экран из файла 'TEX.TXT' вывести список техники. Каждое наименование выводить с новой строки.*

```
Program VV_TEX; {Имя программы}
Var {Раздел описания переменных}
SP:text;
C:char;
i,n:integer;
Begin
Assign(SP,'TEX.TXT'); {подключение файла 'TEX.TXT' к файловой
переменной F}
```

```
Rewrite(SP); {открытие файла на запись}
Write('Введите количество строк в списке:'); {Вывод сообщения}
Readln(n); {вод значений переменной n с клавиатуры}
Writeln('Введите наименование техники ',n:1,' наименований');
For i:=1 to n do {цикл для ввода строк текста с клавиатуры в файл}
Begin
While not eoln do {цикл для ввода одной строки текста в файл}
Begin
Read(C); {ввод текста в переменную C}
Write(SP,C); {запись текста из переменной в файл}
End; {конец внутреннего цикла}
Readln;
Writeln(SP); {запись в файл}
End; {конец внешнего цикла}
Close(SP); {закрытие фала}
Writeln('список техники'); {Вывод сообщения}
Reset(SP); {открытие файла на чтение}
While not eof (SP) DO {внешний цикл чтения строк}
Begin
While not eoln(SP) DO {Внутренний цикл чтения одной строки}
Begin
Read(SP,C); {чтение во вспомогательную переменную}
Write(C) {вывод на экран}
End; {конец внутреннего цикла}
Readln(SP); {чтение из файла}
Writeln; {переход на новую строку}
End {конец внешнего цикла}
End.
```

Самостоятельная работа № 16.

1. Набрать, отладить и выполнить программы, реализующие алгоритмы Вашего индивидуального задания.
2. Взять вариант из самостоятельной работы № 15 и выполнить заполнение массива исходными данными из файла

<*фамилия студента*>_in.txt , а результат записать в файл с именем <*фамилия студента*>_out.txt

3. Составить блок-схему.
4. Проанализировать работу операторов, пользуясь отладочными режимами.
5. Составить краткий конспект. Защитить работу.

Самостоятельная работа основана на заданиях предыдущей самостоятельной работы № 15 и состоит из двух задач.