



Министерство образования
Республики Беларусь

**БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра «Гидротехническое и энергетическое
строительство»**

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

**Лабораторный практикум
по дисциплине «Информатика»**

Минск
БНТУ
2010

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Кафедра «Гидротехническое и энергетическое строительство»

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

Лабораторный практикум
по дисциплине «Информатика»
для студентов специальностей

1-70 04 01 «Водохозяйственное строительство»,
1-70 07 01 «Строительство тепловых и атомных станций»,
1-37 03 02 «Кораблестроение и техническая эксплуатация
водного транспорта»

Минск
БНТУ
2010

УДК 004.42 (076.5)

ББК 32.973-018.2

П 78

Составители:

Г.Н. Дытко, О.Б. Сенько

Рецензенты:

О.Б. Корбут, Е.С. Калиниченко

П 78 Программирование в среде TURBO PASCAL 7.0: лабораторный практикум по дисциплине «Информатика» для студентов специальностей

1-70 04 01 «Водохозяйственное строительство», 1-70 07 01 «Строительство тепловых и атомных станций», 1-37 03 02 «Кораблестроение и техническая эксплуатация водного транспорта» / сост.: Г.Н. Дытко, О.Б. Сенько. – Минск: БНТУ, 2010. – 177 с.

Издание содержит 15 лабораторных работ, краткие теоретические сведения, методические указания, тексты программ-примеров основных разделов: линейные, разветвляющиеся и циклические алгоритмы, подпрограммы, массивы, строки, записи и множества, а также варианты заданий для индивидуальной проработки.

ISBN 978-985-525-361-8

© БНТУ, 2010

ОСНОВЫ ПРОГРАММИРОВАНИЯ

ВВЕДЕНИЕ

Уважаемые студенты!

Вы начинаете изучение системы программирования Turbo Pascal версии 7.0.

Текст, который вы сейчас читаете, поможет вам правильно организовать свои занятия, и в конечном итоге, успешно овладеть новыми для вас знаниями.

Как мы будем работать

Программирование, как и любой другой предмет, можно изучать по-разному. Но наилучшие результаты в изучении языков программирования достигаются при систематической работе с компьютером.

Привычное задание будет всегда индивидуальным. Благодаря такому подходу каждый из вас имеет возможность проявить свою индивидуальность и самостоятельность. Большую часть материала вам придется осваивать самостоятельно. Учебный материал взаимосвязан и переплетен. Практически каждую лабораторную работу просто невозможно выполнить, не выполнив предыдущей лабораторной работы.

За лабораторное занятие вам необходимо:

- разобраться с теоретическим материалом и примерами, при необходимости прибегая к помощи преподавателя;
- законспектировать основные положения лабораторной работы;
- выполнить практические задания по тексту лабораторной работы, и составить программы (начиная с лабораторной работы № 2);
- отчитаться по выполненной работе.

Несмотря на достаточно "демократическую" форму занятий, к вам предъявляется ряд требований.

Ваши обязанности:

1. Вести аккуратный конспект лабораторных работ, теоретической и практической части.
2. Каждую лабораторную работу целесообразно начинать с нового листа.
3. Своевременно отчитываться за каждую лабораторную работу.

4. За каждую лабораторную работу вы должны успеть получить одну оценку за теоретическую часть и оценки за выполнение индивидуальных заданий.

5. Приступать к выполнению лабораторной работы нужно только после полного отчёта за все предыдущие работы.

6. Не пропускайте занятий - навёрстывать упущенное будет тяжело.

Ваши права:

1. Отчитываться за лабораторную работу, в зависимости от сложившихся обстоятельств, опережая график выполнения лабораторных работ.

2. Работать над лабораторной работой дома.

3. Пользоваться учебной, научной и справочной литературой.

Итоговая оценка за выполнение каждой лабораторной работы - выставляется по совокупности качества и своевременности выполнения лабораторных работ.

Лабораторная работа № 1

ИНТЕГРИРОВАННАЯ ИНСТРУМЕНТАЛЬНАЯ ОБОЛОЧКА (ИИО) СИСТЕМЫ TURBO PASCAL 7.0 (TP 7.0)

Цель работы: изучить структуру интегрированной среды программирования Turbo Pascal версии 7.0 (TP 7.0). Изучить структуру программ и правила их записи на языке Turbo Pascal версии 7.0. Приобрести начальные навыки работы в системе Turbo Pascal на примере программирования линейных алгоритмов.

Задачи:

- ◆ научиться работать в интегрированной среде Turbo Pascal версии 7.0 (TP 7.0);
- ◆ копировать, перемещать и удалять фрагменты программы.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Запуск интегрированной среды (ИИО) Turbo Pascal 7.0 (TP 7.0) осуществляет файл TURBO.EXE.

В настоящее время данная система является одной из самых популярных промышленных (профессиональных) систем программирования. Система программирования Turbo Pascal представляет собой интегрированную среду. Все, что мы называем языком программирования TP7, объединено в так называемой интегрированной инструментальной оболочке (ИИО).

Интегрированная среда (Турбо-среда) позволяет набирать тексты программ с использованием встроенного редактора текстов, компилировать их, выполнять, проводить отладку программ.

Она включает в себя:

- многооконный текстовый редактор, для обработки текстов программ на языке TP;
- компоновщик программ;
- отладчик программ;
- система контекстной информационной помощи (на английском языке);
- компилятор (переводчик) текста программ на языке TP в программу на машинных кодах.

Существует два "вида" переводчиков языков программирования,

основная задача которых, перевести программу с языка программирования в машинные коды понятные компьютеру:

а) "Компилятор" переводчик текста программ. Он переводит всю программу сразу и только после этого, если в ней нет ошибок, запускает её на выполнение.

б) "Интерпретатор", переводит текст программы построчно и сразу построчно её выполняет.

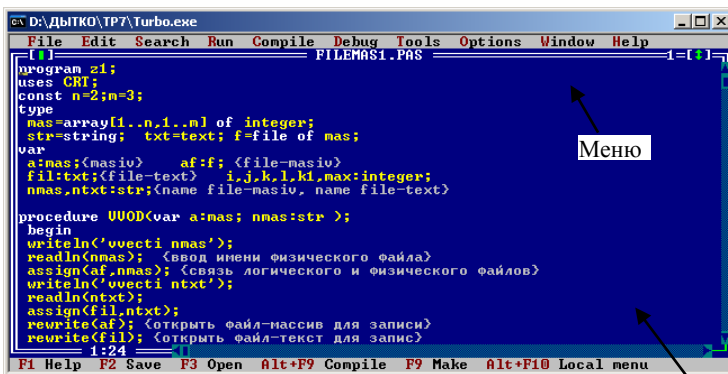
Структура основного экрана ИИО TP 7.0

По внешним признакам интегрированную инструментальную оболочку (ИИО) (рисунок 1) можно разделить на три различные по функциональному назначению области:

1) верхняя часть экрана – "Меню" управления всеми ресурсами ИИО TP 7.0. С помощью Меню можно очень быстро освоить ИИО TP 7.0, так как это "Меню" управляет всеми функциональными возможностями данной системы.

2) средняя часть экрана – область многооконного текстового редактора (или рабочая зона). Именно в этой области размещаются и редактируются тексты программ, или любой другой текст.

3) нижняя часть экрана – строка состояния о некоторых из доступных операций ИИО и комбинации клавиш для их быстрого вызова.



Строка
состояния

Рисунок 1 – Структура ИИО

Окно текстового
редактора

Строка меню

Вход в Меню – обеспечивает нажатие клавиши [F10] или щелчок мыши. Все операции можно выполнить с помощью Меню ИИО. Вернуться из любого места Меню в редактор можно нажав клавишу Esc.

Основные режимы (опции) Меню ИИО ТР:

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

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

➤ Search: позволяет осуществлять поиск фрагментов текста и при необходимости производить замену найденного фрагмента новым;

➤ Run: позволяет запускать программу, находящуюся в рабочей области, а также при необходимости пошагово выполнять данную программу или ее часть;

➤ Compile: позволяет осуществлять компиляцию программы, которая находится в рабочей зоне;

➤ Debug: содержит команды, облегчающие процесс поиска ошибок в программе (Breakpoints – точки останова, окно отладки Watch, окно используемых подпрограмм, окно регистров, окно выходных результатов и некоторые другие);

➤ Tools: позволяет выполнять некоторые подпрограммы, не выходя из ИИО ТР 7.0;

➤ Options: позволяет установить необходимые для работы параметры компилятора и ИИО ТР 7.0;

➤ Window: позволяет выполнять все основные операции с окнами (открывать, закрывать, перемещать, изменять размер);

➤ Help: позволяет получить имеющуюся в системе справочную информацию.

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

Кроме подчиненных меню, существует так называемое локальное (контекстное) меню, которое содержит наиболее часто используемые команды из меню и некоторых других. Открыть контекстное меню можно посредством комбинации клавиш [Alt+F10] или при помощи правой кнопки мыши.

Рабочая зона

По мигающему курсору можно определить, где Вы работаете в данный момент в активном окне – либо в Меню, либо в рабочей зоне (экранном текстовом редакторе). Курсор, можно передвигать в активном окне с помощью клавиш управления курсором (←↑→↓), либо мыши.

Активным называется окно редактора, ограниченное двойной, белой рамкой. На этой рамке размещаются специальные символы для управления этим окном. В правом верхнем углу размещается номер окна. Окна нумеруются, начиная с единицы. В новом окне в середине верхней части рамки есть надпись: NONAME00.PAS или NONAME01.PAS и т.д., в зависимости от количества открытых окон. NONAME в переводе файл без имени. Так в середине верхней части рамки окна рисунка 1 можно прочесть имя файла *FILE MAS1.PAS*, имя которого дано было пользователем при сохранении текста программы на диске.

Для работы в редакторе можно использовать любое количество окон. Любое окно можно 1) открыть, 2) закрыть, 3) сделать активным окном, 4) изменить его размеры и положение на экране. Активным может быть только одно из всех открытых окон. В середине верхней части рамки размещается имя файла, помещённого в окне для редактирования.

Справа и снизу окна расположены так называемые полосы прокрутки (скроллинга). В каждой полосе расположен свой "курсор", который указывает на текущее положение курсора окна относительно текста. Таким образом, взглянув на полосу скроллинга, сразу можете определить, в каком месте текста находитесь. Кроме того, передвижение окна по тексту вверх/вниз на одну страницу может осуществляться с помощью клавиш [PgUp]/[PgDn], а на одну строку – клавишами управления курсором. В левом верхнем углу находится кнопка закрытия окна, которая находится в его левом верхнем углу, а

также с помощью комбинации клавиш [Alt+F3] или соответствующей команды, выбранной в меню Window.

Строка состояния

Строка состояния или строка подсказок пользователю (программисту) находится в нижней части экрана о состоянии системы и действиях доступных в данный момент, а также комбинации клавиш для их быстрого вызова, которые позволяют выполнить соответствующие команды, минуя стандартную процедуру их вызова через меню.

Создание программ в ИИО TP 7.0

Состав программы

Pascal-программа состоит из заголовка, раздела описаний (объявлений) и раздела операторов

Program имя_программы;

Раздел описаний

Begin

Раздел операторов

End.

Управление конфигурацией

Перед началом написания программ на языке Turbo Pascal 7.0 (TP 7.0) необходимо провести (или проверить) установку опции конфигурации Options. Кроме того, необходимо позаботиться о том, чтобы при каждом вызове TP 7.0 автоматически восстанавливались ранее выбранные Вами опции и последний из обрабатываемых файлов загружался в текущее окно для обработки редактором текста.

Сначала открываем меню Options (например, с помощью горячих клавиш [Alt+O]). Установка параметров ИИО производится посредством дополнительного меню Environment, которое содержит следующие элементы:

- Preferences... — позволяет установить параметры, определяющие условия работы ИИО в целом;

- Editor... — позволяет установить параметры, определяющие условия работы редактора ИИО;
- Mouse... — позволяет установить опции, определяющие работу манипулятора мышь;
- Startup... — позволяет установить опции, определяющие начальные параметры ИИО;
- Colors... — позволяет установить желаемый цвет для каждого из элементов ИИО.

В результате выбора команды Options на экране появляется окно переустановок, изображенное на рисунке 2. Окно переустановок можно разделить на 5 областей. Нас интересует область Auto Save (автоматического сохранения). Перемещение между областями осуществляется с помощью клавиши [ТАВ] или мыши. Смена значений выбранной области (установка или отмена) производится нажатием клавиши [Пробел]. Чтобы выбранная Вами конфигурация ИИО TP 7.0 сохранилась, ее необходимо записать на диск. Для этого в меню Options выберете команду Save As... меню Options, ввести свое имя файла с расширением TP (вместо TURBO.TP). Чтобы сохранить конфигурацию среды в уже существующем файле, необходимо воспользоваться командой Open меню Options. В открывшемся окне выберите файл конфигурации среды из уже существующих файлов, список которых находится в поле File.

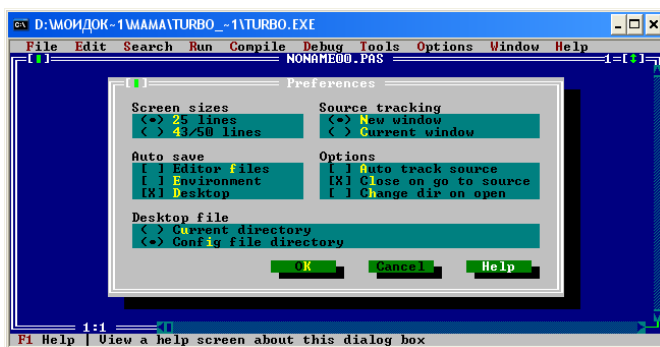


Рисунок 2 – Окно переустановок Preferences

Ввод и корректировка программы

Установив необходимые опции, если это было необходимо, возвратитесь в рабочую зону, воспользовавшись клавишей [ESC]. В появившемся активном окне NONAME00.PAS текстового редактора можно ввести текст программы. Мигающий курсор указывает то место на экране, в котором будет появляться текст. Набирая текст, особое внимание обращайтесь на точное воспроизведение всех знаков.

Всем работающим с этим языком программирования приходится использовать возможности операций над фрагментами текста любого размера, или иначе - "блоками".

Для выделения блока, необходимо подвести курсор к началу необходимого фрагмента и удерживая клавишу [Shift] передвигать курсор к концу этого фрагмента. Выделенный фрагмент инверсно выделяется на экране (серым цветом).

Снять сделанное ранее выделение фрагмента текста можно таким же образом. Необходимо подвести курсор к уже выделенному фрагменту текста, и с помощью клавиши [Shift] уменьшить размер выделенной области до "нуля", либо щелкнуть левой кнопкой мыши вне выделенного блока.

Редактирование текста программы

Система TP 7.0 позволяет выполнять следующий ряд операций над выделенными фрагментами текста с помощью меню Edit:

- Undo: возвращает на несколько шагов обратно, и таким образом отказаться от последних изменений
- Redo: отменяет действия, выполненные операцией Undo
- Cut: вырезает выделенный фрагмент в "карман". В текущем тексте выделенный фрагмент будет удалён.
- Copy: копирует содержимое выделенного фрагмента в "карман" без его удаления из текста.
- Paste: вставляет текст, из "кармана" начиная с текущей позиции курсора.
- Clear: удаляет выделенный фрагмент (без записи его в "карман").
- Show clipboard: активизирует окно Clipboard ("промежуточный буфер" или "карман").

Сохранение программы на диске

Перед исполнением созданную программу необходимо сохранить. Для этого воспользуйтесь строкой подсказки (клавиша [F2]), при нажатии которой выполняется операция Save (Сохранение). В открывшемся диалоговом окне Save File As задайте имя программы и нажмите клавишу [Enter]. Расширение PAS в имени файла добавляется TP 7.0.

Примечание: Созданную программу можно сохранить, вызвав команду Save или Save As меню File. Команда Save позволяет записать файл под прежним именем, а команда – Save As под другим выбранным именем.

Выполнение программы

Компиляция и запуск программы в TP 7.0 осуществляется комбинацией клавиш [Ctrl+F9] (или вызовом команды меню Run из меню Run). Если транслятор обнаружит в программе синтаксическую ошибку. Он выдает соответствующее сообщение (Error...) на экран, указав место ошибки. Необходимо устранить все ошибки, проверив тщательно синтаксис написания программы на TP 7.0.

Примечание: для компиляции программы без ее непосредственного запуска воспользуйтесь комбинацией клавиш [Alt+F9]. Откомпилированный файл будет сохранен по умолчанию в памяти, если указано в области Destination значение Memory или на диск, если значение – Disk.

Для запуска оттранслированной программы необходимо воспользоваться комбинацией клавиш [Ctrl+F9]. Просмотреть результат работы программы, т.е. выводимую на экран монитора информацию, а не сам текст программы, можно благодаря режиму "разумного переключения".

Для этого необходимо выполнить следующее:

1. Открыть меню Debug.
2. Вызвать команду User Screen (экран пользователя) (можно с помощью комбинаций клавиш для быстрого вызова – [Alt+F5]).
3. Активизировать экран, нажав клавишу [Enter] или воспользоваться комбинацией клавиш [Alt+F5];
4. Завершить работу, нажав любую клавишу, при этом Вы вернетесь в первоначальное окно, из которого вызван был этот режим.

Примечание: Если в процессе запуска программы возникает необходимость прервать работу (например, в случае "зацикливания" программы), воспользуйтесь комбинацией клавиш [Ctrl+Break].

Идентификаторы

Имена (или идентификаторы) вводятся программистом для обозначения в программе переменных, констант, типов, меток, процедур, функций, модулей и т.п. Идентификатор должен удовлетворять следующим условиям:

- начинаться с латинской буквы (TP 7.0 не различает прописные и строчные буквы) или с символа подчёркивания (_);
- начиная со второй позиции в идентификаторе можно менять наряду с буквами латинского алфавита цифры от 0 до 9;
- пробел в TP является разделителем и не может присутствовать внутри идентификатора. Для создания идентификатора, состоящего из нескольких слов, можно воспользоваться заглавными буквами (*ReadText*) либо символом подчёркивания (*Read_Text*), но не пробелом (*Read Text*);
- зарезервированные слова (такие как *begin*, *end*, *program* в качестве идентификатора не допускаются;
- служебные слова (такие как *write*, *read*, *case* и т.д.) в качестве идентификатора не желательно использовать;
- max длина идентификатора при сравнениях 63 символа.

Справочная система Turbo Pascal 7.0

Справочная система разбита на несколько подразделов, список которых представлен в меню HELP:

- Contents: вызывает экран содержимого справочной системы;
- Index: выводит список ключевых слов, по которым имеется информация в справочной системе;
- Topic search: выводит справочную информацию о термине, на котором расположен курсор;
- Previous topic: выводит содержимое предыдущего окна информационной помощи;
- Using help: выводит справочные сведения о системе контекстной помощи;

- Files: позволяет подключить к системе Help другие справочные файлы;
- Compiler directives: выводит список директив компилятора;
- Procedures and functions: выводит список процедур и функций Borland Pascal;
- Reserved words: выводит список зарезервированных слов;
- Standart units: выводит список стандартных модулей;
- Turbo Pascal Language: выводит список основных элементов TP 7.0;
- Error message: выводит справочную информацию об ошибках;
- About: выводит справочную информацию о самом пакете TP 7.0.

Завершение работы с ИИО TP 7.0

Завершить работу с ИИО TP 7.0 можно с помощью комбинации клавиш [Alt+X] или команды **Exit** меню File.

При необходимости временного выхода из ИИО TP 7.0 в MS DOS необходимо вызвать команду Dos Shell из меню File. При этом ИИО TP 7.0 останется в памяти, но управление передается DOS. После выхода из ИИО можно ввести команды MS DOS или запустить другие программы. Для возвращения в ИИО, нужно набрать команду EXIT в командной строке и нажать клавишу [Enter]. При этом ИИО появится на экране в том же состоянии, в котором была до выхода из нее.

Основы построения программ на TP 7.0

Алфавит языка и специфика использования символов

Текст Pascal-программы представляет собой запись на языке программирования алгоритма решения поставленной задачи. При записи текста используется свой алфавит, при помощи которого образованы величины, выражения и операторы данного языка. Алфавит языка TP 7.0 включает в себя все символы, представленные в кодировочной таблице, которая загружена в оперативную память или хранится в ПЗУ (постоянно запоминающее устройство) компьютера. Каждому символу алфавита соответствует порядковый номер (числовой код) от 0 до 255. Символы с кодами от 0 до 127 представляют собой основную таблицу кодов ASCII.

В качестве символов языка TP 7.0 используются: При записи текста программы используются следующие символы:

- прописные или строчные буквы латинского алфавита от A до Z;
- цифры от 0 до 9;
- специальные символы, выполняющие определенные функции при построении различных конструкций языка:
 - + - * / [] () { } . , ; : ' # \$ _ ^
 - составные символы – группа символов, которые воспринимаются компилятором как единое целое:
 - := <= <> >= .. (* *) (. .)
 - зарезервированные (служебные) слова: and array begin case const div do и т.д.
 - буквы русского алфавита и другие символы могут использоваться только в комментариях, операторах вывода Pascal-программы.

Структура программы на языке Turbo Pascal

Любая программа на языке TP состоит из трех разделов: заголовка программы, раздела описаний и раздела операторов – основной блок программы. Не все разделы обязательны.

1. Заголовок программы состоит из зарезервированного слова PROGRAM, **пробела**, имени данному программе и символа ";", например PROGRAM Hello;

2. Раздел описаний предназначен для объявления данных и может включать в себя подразделы: описания меток, констант, типов, переменных, а также подпрограмм, реализуемых в виде процедур или функций. Каждый из подразделов раздела описаний начинается своим ключевым словом. Раздел меток начинается с ключевого слова, раздел констант - CONST, раздел типов - TYPE, раздел переменных – VAR. Не все подразделы обязательны.

Если в программе используются стандартные или библиотечные модули (UNIT), то первой (после заголовка программы) должна стоять директива USES, в которой перечисляются используемые имена модулей, например: USES CRT; - подсоединяет стандартный модуль CRT, содержащий описание процедур, функций, констант, типов и переменных, позволяющих работать с цветом, звуком, экраном и ускорить операции ввода-вывода данных.

3. Раздел операторов (тело программы) начинается служебным словом `BEGIN` и заканчивается словом `END`. (**END с точкой**), которая является признаком конца программы, иначе говоря, тело программы заключено в операторные скобки (`BEGIN` и `END`), внутри которых содержится последовательность операторов программы.

Примечание: Текст, следующий за оператором `END`., транслятор игнорирует. Для Pascal-программ характерна ступенчатая форма записи. Строки, относящиеся к одной конструкции или связанные по смыслу, записываются с одной и той же позиции. Строки, относящиеся к подчиненной конструкции, записываются правее, например, на две-три позиции, благодаря чему наглядно представляется структура программы. Обычно слова `PROGRAM`, `BEGIN` и `END` записываются с первой позиции.

Любая программа состоит из операторов (предложений) языка, которые располагаются в строках. Операторы отделяются друг от друга символом точка с запятой (;), которая показывает компилятору, где заканчивается один оператор и начинается следующий. Зарезервированное слово `BEGIN`, с которого начинаются блоки программы, не требует после себя символа точка с запятой, и так же не обязательно ставить точку с запятой перед зарезервированным словом `END`.

Запись операторов в строке может начинаться с любой позиции. В одной строке можно записать несколько операторов. Один оператор может быть записан в нескольких строках.

При написании Pascal-программы применяется ступенчатая форма записи. Строки, относящиеся к одной конструкции или связанные по смыслу, записываются с одной и той же позиции. Строки, относящиеся к подчиненной конструкции, записываются правее, например на две-три позиции, благодаря чему наглядно представляется структура программы. Обычно служебные слова **Program**, **Begin** и **End** пишутся с первой позиции.

Объявление меток. Любой оператор может быть помечен меткой – целым десятичным числом (от 0 до 9999), строку символов, символьно-цифровую конструкцию. Метки ставят не на все операторы, а только на те, на которые будут ссылки в программе. Назначение меток: они дают возможность обращаться к нужной строке

PROGRAM имя_программы;	{Заголовок программы}
USES CRT;	{Подключаемые библиотеки}
{ Раздел описаний (объявлений) }	
LABEL	{Описание меток}
CONST	{Описание констант}
TYPE	{Описание типов}
VAR	{Описание переменных}
PROCEDURE (FUNCTION)	{Описание текстов подпрограмм — процедур и функций}
{Раздел основного блока программы — раздел операторов}	
BEGIN	
Оператор_1;	
Оператор_2;	
. . . ;	
Оператор_n;	
END.	

программы из любого места этой программы.

При использовании меток в программе необходимо учитывать следующее:

- а) в одной программе не может быть двух одинаковых меток;
- б) номера меток ставятся в произвольном порядке.

Пример: LABEL 1, 5, 9999, h2, 4t32e, metka_1;

Лабораторная работа № 2

АЛГОРИТМ ЛИНЕЙНОЙ СТРУКТУРЫ

Цель работы: изучить структуру и правила записи программ на языке Turbo Pascal; приобрести начальные навыки работы в системе Turbo Pascal на примере программирования линейных алгоритмов.

Задачи:

- ◆ приобрести практические навыки по организации ввода-вывода стандартных типов данных;
- ◆ научиться строить алгоритм и писать программы линейной структуры.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Константы и переменные

При решении любой задачи требуются данные, над которыми выполняются определенные действия для получения конечного результата. Каждая величина, используемая в программе, может быть либо константой, либо переменной. В программе каждый элемент данных является либо константой, либо переменной. Константы и переменные определяются идентификаторами, по которым к ним обращаются для получения текущих значений.

Константа – величина, значение которой задается в тексте программы в явном виде и в дальнейшем не изменяется.

Формат:

Const

<идентификатор> = <значение константы>;

Символьное или строковое значение константы записывается в апострофах.

Пример:

Const

Max=100\$

Argument=34.5; symbol="!";

Str1='Сегмент1'; stroka='Turbo Pascal';

Переменная – это величина, значение которой может изменяться в процессе выполнения программы.

Каждая величина, будь то константа или переменная, должна относиться к одному из типов данных. Тип переменных должен быть описан перед тем, как с переменными будут выполняться какие-либо действия.

Формат:

Var <идентификатор> : <тип>;

В TP 7.0 допускается использование чисел, представленных в десятичной или шестнадцатеричной системах. Если число представлено в шестнадцатеричной системе, перед ним без пробела записывается знак \$. Диапазон изменений от \$0000 до \$FFFF.

В десятичной системе числа могут записываться двумя способами: с фиксированной и с плавающей точкой.

Вещественные десятичные числа с фиксированной точкой записываются по обычным правилам арифметики. Целая часть от дробной части отделяется десятичной точкой. Если десятичная точка отсутствует, число считается целым. Перед числом может находиться знак "+" или "-". Если знак "+" отсутствует, по умолчанию число считается положительным.

Пример:

- 125 – целое десятичное число
- \$1FF – шестнадцатеричное число
- 124.34 – вещественное (действительное) число

Вещественные десятичные числа в форме с плавающей точкой представлены в экспоненциальной форме $mE+p$, где m – мантисса (целое или дробное с десятичной точкой), "E" означает десять в степени", p – порядок (целое число).

Пример:

$$5.18E+02 = 5.18 \cdot 10^2 = 518$$
$$10Y-03 = 10 \cdot 10^{-3} = 0.01$$
$$3.14E00 = 3.14 \cdot 10^0 = 3.14$$

Комментарий

Комментарии – произвольная последовательность символов, в том числе и русских букв, заключенных в фигурные скобки {...} или (* ... *), предназначенная для пояснений в программе. Комментарий игнорируется компилятором и поэтому на программу не оказывает никакого влияния.

Пример:

```
{ Комментарий к программе Regres  
*****}  
(*Программа вычисления корня нелинейного уравнения*)
```

Типы данных

Тип – это множество значений, которые могут принимать константы, переменные, функции, выражения.

TP 7.0 наряду со стандартными типами (целочисленный, вещественный, символьный, булевский), позволяет образовывать программисту собственные типы.

Все допустимые типы в TP 7.0 подразделяются на две большие группы: простые (скалярные и пользовательские) и структурированные. Простые – подразделяются на порядковые и вещественные. Структурированные типы базируются на порядковых типах и могут содержать их различные комбинации.

Порядковые типы отличаются тем, что каждый из них имеет конечное число возможных значений, поэтому каждому из этих значений можно сопоставить некоторое целое число – порядковый номер значения.

Вещественные типы тоже имеют конечное число значений, которое определяется форматом внутреннего представления вещественного числа. Однако количество возможных значений так велико, что сопоставить с каждым из них порядковый номер не представляется возможным.

Порядковые типы

К порядковым типам относятся целые, логический, символьный, перечислимый и тип–диапазон.

Целые типы – множество целых чисел в различных диапазонах, зависящих от внутреннего представления, занимают в памяти компьютера один, два или четыре байта.

Название	Длина, байт	Диапазон значений
Byte	1	0...255
ShortInt	1	-128...+127
Word	2	0...65535
Integer	2	-32768...+32767
LongInt	4	-2 147 483 648...+2 147 483 647

Пример:

Var

A : integer; d34, Done : word;
x:byte; Ntel:longint;

Логический тип данных (BOOLEAN) может принимать лишь два значения: *FALSE* (ложь) и *TRUE* (истина).

Пример:

```
Var  
x:Boolean;
```

Символьный тип данных (CHAR) – предназначен для хранения одиночных символов. Каждому символу ставится целое число 0...255. Это число служит кодом внутреннего представления символа. Для кодировки используется код таблицы ASCII (American Standart Code for Information Interchange – американский стандарт код для обмена информацией). Первая половина символов с кодами 0...127 соответствует стандарту ASCII. Вторая половина символов с кодами 128...255 не ограничена жесткими рамками стандарта и может меняться на ПК разных типов. Символы с кодами 0...31 относятся к служебным кодам.

Пример:

```
Const  
X='?'; Y='1'; Z='+';  
Var Znak, Simvol:char;
```

Перечислимый тип данных задается списком значений (объектов), которые могут принимать переменные этого типа/

Пример:

```
Type  
Year=(jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec);  
Colors=(red, white, blue);
```

```
Var  
month: Year;  
col:Colors;
```

Переменные любого перечислимого типа можно объявлять без предварительного описания этого типа

Пример:

```
Var  
month: (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec);  
col: (red, white, blue);
```

Ограниченный тип данных (тип-диапазон) задается границами своих значений внутри базового типа:

<минимальное значение>..*максимальное значение*>;

Здесь – <минимальное значение> – минимальное значение типа-диапазон;

– <максимальное значение> – максимальное значение типа-диапазон.

Пример:

Type

date=1..31;

Month=1..12; Simvol='A'..'Z';

Var

Data:date; mes:Month;

или

Var

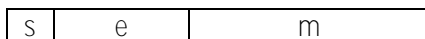
date:1..31;

Month:1..12; Simvol='A'..'Z';

Вещественные типы данных представляют произвольные числа лишь с некоторой точностью, зависящей от внутреннего представления формата вещественного числа. TP поддерживает несколько различных вещественных типов.

Длина, байт	Название	Количество значащих цифр	Диапазон десятичного порядка
6	Real	11...12	-39...+38
8	Double	15...16	-324...+308
10	Extended	19...20	-4951...+4932
8	Comp	19...20	$-2 \cdot 10^{63} + 1 \dots -2 \cdot 10^{63} - 1$

Занимает от 4 до 10 байт и имеет следующую структуру в памяти ПК:



Здесь s – знаковый разряд числа; e – экспоненциальная часть; содержит двоичный порядок; m – мантисса числа

Данные вещественного типа могут быть представлены в двух видах: с фиксированной точкой и с плавающей точкой:

Значение	С плавающей точкой	С фиксированной точкой
0	0.0000000000E+00	0
134	1.3400000000E+02	134
23.545	2.3545000000E+01	23.545
-75.11	-7.5110000000E+01	-75.11

Пример:

Var

Rezult: real; ltog,summat: extendet;

X,Y : real;

Структурированные типы

Структурированные типы данных определяют упорядоченную совокупность порядковых переменных и характеризуются типом своих элементов (компонентов). К ним относятся следующие типы данных: строки, массивы, множества, записи, файлы и указатели. Эти типы подробно будем рассматривать в последующих лабораторных работах. Здесь приводится лишь их краткая характеристика.

Строка – это последовательность символов, заключенная в апострофы. Определение строкового типа устанавливает максимальное количество символов, которое может содержать строка и начинается зарезервированным словом **STRING** (строка). Размер строки может изменяться от нуля до максимально заданной при описании величины. Количество символов, которое содержит данная строка, указывается в квадратных скобках.

Пример:

Const

Stroka='Программирование на языке Turbo Pascal '

Print='HP LaserJet 1200 – печатающее лазерное устройство'

Var

Str:String; {длина строки максимальная – 255 символов}

Slovo:string[15]; {длина строки максимальная – 15 символов}

Массив – это структурированный тип данных, состоящий из определенного числа элементов, имеющих один и тот же тип. Каждый элемент массива определяется индексом, по которому к нему осуществляется доступ. Число элементов фиксируется при описа-

нии и в процессе выполнения программы не изменяется. Для определения массива служит зарезервированное слово **ARRAY** (массив).

Множество это структурированный тип данных, представляющий набор выбранных по какому-либо признаку или группе признаков объектов, которые можно рассматривать как единое целое. Множество задает интервал значений, который является множеством всех подмножеств. Элементы множества не упорядочены. Для описания множества используется зарезервированное слово **SET** (множество).

Запись – это структурированный тип данных, состоящий из фиксированного числа компонент разного типа. Определение типа записи начинается зарезервированным словом **RECORD** (запись) и задает для каждого компонента, называемого полем, его тип и обозначающий это поле идентификатор.

Файл – это структурированный тип данных, состоящий из последовательности компонент одного типа и одной длины. Чаще всего компонентами являются записи. Для определения файла используется зарезервированное слово **FILE** (файл). Файл хранится на диске и вызывается в оперативную память для обработки по мере необходимости.

Выражения, операнды, операции

Переменные и константы, обращение к функциям – это основные элементы из которых конструируются программы. Каждый из этих элементов характеризуется своим значением и принадлежит к какому-либо типу данных.

Выражение задает порядок выполнения действий над элементами данных и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций. Операции определяют действия, которые надо выполнить над операндами. Операции в языке TP 7.0 подразделяются на арифметические, отношения. Логические (булевские), строковые и т.д. Выражения соответственно называются арифметическими, отношения, булевскими, строковыми и т.д., в зависимости от того, какого типа операнды и операции в них используются.

При написании арифметических выражений во избежание ошибок, а также для задания очередности выполнения арифметических операций, целесообразно использовать скобки, причем количество открытых скобок должно быть равно количеству закрытых скобок.

Пример:

$a \text{ div } b$, если $a=10$ и $b=3$ то результат = 3

$a \text{ mod } b$, если $a=10$ и $b=3$ то результат = 1

Записать на алгоритмическом языке Pascal арифметическое

выражение $\frac{x \cdot y}{x^2 + 7.5} + \cos x$

Решение: $X*Y/(SQR(X)+7.5)+COS(X)$;

В приведенном примере происходит обращение к библиотечным функциям Pascal **SQR**, **COS**, полный список которых содержится в прил. А. При обращении к библиотечным функциям необходимо указать имя функции, а после имени в скобках - аргументы. При этом аргументы могут быть арифметическими выражениями.

При использовании библиотечных функций необходимо учитывать следующее:

➤ отрицательное число не может быть возведено в вещественную степень;

➤ для тригонометрических функций аргумент указывается в радианах.

Операции отношения выполняют сравнение двух операндов и определяют истинно значение или ложно: < (меньше), > (больше), <= (не больше, т.е. меньше и равно), >= (не меньше, т.е. больше и равно), = (равно), <> (не равно).

Пример:

$A >= B$ при $A=12$ и $B=12$ ($12 >= 12$) – результат равен True

$C > D$ при $C=34$ и $D=10$ ($34 > 10$) – результат равен True

$A <= B$ при $A=11$ и $B=6$ ($11 <= 6$) – результат равен False

Логические выражения и операции применяются к величинам логического типа (т.е. типа *Boolean*), результат операции тоже логического типа: **AND** – И; **OR** – ИЛИ; **XOR** – исключаящее ИЛИ; **NOT** – ОТРИЦАНИЕ и к операндам целого типа, результат – тоже целое число, биты которого (двоичные разряды) формируются из

битов операндов по определенным правилам.

Пример:

выражение `a<10 and B<100` записано не верно, необходимо записать `(a<10) and (B<100)`.

Операторы

Основная часть программы на языке Turbo Pascal представляет собой последовательность операторов, каждый из которых производит некоторое действие над данными. Все операторы подразделяются на три группы: простые, ввода-вывода и структурные, кроме трех групп отдельно выделяется оператор `With`.

Простые операторы

Простые – операторы, не содержащие в себе никаких других операторов. К ним относятся оператор присваивания, безусловного перехода, вызова процедуры и пустой оператор.

Оператор присваивания `<Идентификатор> := <Выражение>;`

Слева в операторе присваивания всегда стоит имя переменной, а справа – то, что представляет собой её значение (конкретное значение, арифметическое или логическое выражение, вызов функции, либо имя другой переменной). Переменная и выражение должны иметь один и тот же тип (кроме случая, когда переменная имеет вещественный тип, а выражение – целочисленный). Допустимо присваивание любых типов данных, кроме файловых.

Пример:

```
sort:=1; Cena:=15,23; Naimen:='Грунт - песок';  
Rezult:=Cos(x)+Sin(y);
```

Оператор безусловного перехода (`Goto`) имеет вид: `Goto <Метка>;` означает "перейти к" и применяется в случаях, когда после выполнения некоторого оператора необходимо выполнить не следующий по порядку, а какой-либо другой, отмеченный меткой оператор. Метка может содержать как цифровые, так и буквенные символы.

Пример:

```
Label Metka1, Metka2;  
Metka1:Goto Metka2;  
Metka2:Goto Metka1;
```

При записи оператора **Goto** необходимо помнить следующее:

- метка, на которую передается управление, должна быть описана в разделе меток того блока процедуры, функции, основной программы, в котором эта метка используется;
- областью действия метки является тот блок, в котором она описана; переход возможен только в пределах блока;
- попытка выйти за пределы блока или передать управление внутри другого блока вызывает программное прерывание.

Пустой оператор не содержит никаких символов и не выполняет никаких действий. Он может быть расположен в любом месте программы. Где синтаксис языка допускает наличие оператора. Пустой оператор может быть помечен меткой. Чаще всего оператор используется для организации выхода из середины программы или составного оператора.

Пример:

```
Begin  
    Goto Metka;                {переход в конец блока}  
    ...  
    Metka:                    {пустой оператор помечен меткой}  
End.
```

Операторы ввода-вывода

Решение самой простой задачи на ПК не обходится без операций ввода-вывода информации. Ввод данных – это передача информации от внешнего носителя в оперативную память для обработки. Вывод – обратный процесс, когда данные передаются после обработки из оперативной памяти на внешний носитель.

Для выполнения операций ввода-вывода служат четыре оператора: **read**, **readln**, **Write**, **Writeln**.

Операторы ввода (чтения) данных обеспечивают ввод числовых данных, символов, строк и т.д. для последующей обработки программой: `Read (Readln)` (список ввода);

1) `Read(x1, x2, ..., xn)`; где x_1, x_2, \dots, x_n – переменные допустимых типов данных.

Значения x_1, x_2, \dots, x_n набираются пользователем с клавиатуры минимум через один пробел во время выполнения программы и высвечиваются на экране монитора. После набора данных для одного оператора `Read` нажимается клавиша `[Enter]` (ввод). Значения переменных должны вводиться в строгом соответствии с типом данных, которые объявлены до работы с ними.

Пример:

```
VAR
  A:integer; B:real; C:char;
Begin
  read(A, B); read(C);
End.
```

Первый вариант ответа:

187 1.54 '!'

Нажать клавишу `[Enter]`

Второй вариант ответа:

'!' 187 1.54

Нажать клавишу `[Enter]`

Первый вариант обеспечивает нормальный ввод данных, так как значения `187 1.54 '!'` соответствуют типам переменных `A, B, C` в операторе `read`. Второй вариант ввода вызовет ошибку с кодом 10, т.к. для переменной `A` типа `integer` набирается значение типа `char`.

Если в программе несколько операторов `read`, данные для них вводятся потоком, т.е. после считывания значений переменных для одного оператора `read` данные для следующего оператора `read` набираются в той же строке, а затем происходит переход на новую строку.

Пример:

```
VAR
  A, B, Sum1: integer;
  C, D, Sum2: real;
Begin
  read(A, B); Sum1:=A+B;
  read(C,D); Sum2:=C+D;
End.
```

После набора каждой пары данных нажимаем клавишу ввода [Enter], т.е.

187 32 [Enter], 2.6E-02 1.45E+01 [Enter].

Оператор `read` можно использовать для организации пауз произвольной длины при выполнении программы. Для этого достаточно записать: `read(Ch); {Ch – символьная переменная}`. Программа продолжит работу только после того, как будет нажата любая клавиша на клавиатуре.

2) Оператор `readln` аналогичен оператору `read`. Единственное отличие в том, что после считывания последнего в списке значения для одного оператора `readln` данные для следующего оператора `readln` будут считываться с новой строки: `readln(x1, x2, ..., xn);` где `x1, x2, ..., xn` – переменные допустимых типов данных.

Заменяем операторы `read` на `readln` в предыдущем примере:

```
VAR
  A, B, Sum1: integer;
  C, D, Sum2: real;
Begin
  readln(A, B); Sum1:=A+B;
  readln(C,D); Sum2:=C+D;
End.
```

Набираем на клавиатуре:

187 32 [Enter],
2.6E-02 1.45E+01 [Enter].

3) Пустой оператор `readln;` – происходит переход на новую строку без ввода данных.

Операторы вывода (записи) данных. Для вызова процедуры вывода используются три оператора вывода данных:

1) `write(список ввода);` т.е. `write(x1, x2, ..., xn);` где `x1, x2, ..., xn` – выражения типа `integer, byte, real, char, boolean` и т.д. выводит последовательно значения переменных из списка (числовые данные, символы, строки, булевские значения, арифметические и логические выражения и т.д.)

2) `writeln(список ввода); writeln (x1, x2, ..., xn);` – то же, что и оператор `write` но после вывода переменных осуществляется пере-

ход на новую строку (следующий оператор вывода будет выводить данные с начала новой строки).

3) `writeln`; – осуществляет переход на новую строку без вывода данных.

Как и при выводе последовательно расположенные операторы вывода 1, 3 эквивалентны одному оператору 2.

Чтобы вывести значение переменной, необходимо в операторе вывода `write` в скобках просто указать **имя этой переменной**. Для вывода текстовых сообщений, необходимо в операторе вывода `write` в скобках взять в одинарные кавычки (**апострофы**) это сообщение.

Пример:

```
write(x1);           {выводит значение переменной x1 }
```

```
{выводит текст: Значение переменной x1 равно }
```

```
write('Значение переменной x1 равно ');
```

```
{выводит на принтер: Результат = <значение переменной> rez }
```

```
write(LST,'Результат = ',rez);
```

Форматы вывода данных

В процедурах вывода имеются две возможности выводить данные: без указания ширины поля вывода (*бесформатный вывод*) и с указанием ширины поля вывода (*форматный вывод*).

Бесформатный вывод

Целые, символьные и логические – выводятся, начиная с позиции курсора

Пример:

```
a=15           write('A=',A); → A=15
```

```
c='X'         write('C=',C); → C=X
```

```
d=True       write('D=',D); → D=True
```

Вещественные – выводятся в поле шириной 17 позиций в формате с плавающей точкой. Дробная часть мантиссы содержит 10 цифр.

Пример:

```
a=125.286     write('A=',A); → A=1.2528600000E+02
```

```
b=-2.281e1    write('B=',B); → B=-2.2810000000E+01
```


Форматный вывод

Общий вид $E:m$, где E – имя переменной, значение которой выводится на экран, а m – ширина поля вывода в позициях.

Целые – выводятся в правые крайние аппозиции поля шириной m , где " " – пробел.

Пример:

```
a=17      write('A=',A:5);      → A=___17
b=3456    write('B=',B:5);      → B=_3456
```

Вещественные – выводятся в крайние правые аппозиции поля шириной m в формате с плавающей точкой. Минимальная ширина поля равна 8, в противном случае она игнорируется.

Пример:

```
a=134.25  write('A=',A:11);      → A=1.3425E+02
b=-134.25 write('B=',B:11);      → B=-1.3425E+02
```

В случае, если форматный вывод имеет вид $E:m:n$, где n число позиций дробной части, то значение переменной E выводится в виде числа с фиксированной точкой.

Пример:

```
r=3.1743  write('R=',R:5:2);      → R=_3.17
k=1.25    write('K=',k:7:3);      → K=___1.250
```

Если выводимое данное имеет меньше знаков, чем m то оно дополняется слева пробелами, если больше, то выводится столько знаков, сколько необходимо для корректного представления результата.

Перечень стандартных процедур и функций, применимых к целым, вещественным, символьным и логическим типам представлен в приложении А.

Структурные операторы

Структурные операторы представляют собой структуры, построенные из других операторов по строго определенным правилам. Все структурные операторы подразделяются на три группы: составные, условные и повтора.

Составной оператор – это объединение нескольких операторов в одну группу или блок, отделенных друг от друга точкой с запятой и ограниченное операторными скобками **Begin** и **End**.

```
Begin
  <Оператор_1>;
  ...
  <Оператор_n>;
End;
```

Пример:

```
Begin
  A:=A*B+(C-D);
  Rez:=A;
  Writeln(Rez:3:1);
End;
```

Составной оператор воспринимается как единое целое и может находиться в любом месте программ, где синтаксис языка допускает наличие оператора.

Условные операторы и операторы повтора будут рассмотрены в последующих лабораторных работах.

Программирование линейного алгоритма

Задача 1: составить программу для вычисления значений функций Y и F

$$Y = e^{-ax} \cdot (x \cdot \sin(ax + b) - \sqrt{x} \cdot \cos(bx));$$

$$F = \ln \sqrt{|ax^2 + b|} - 1$$

для заданных значений переменной x и постоянных a и b . Значения переменной $x >= 0$. Включить в программу комментарии. Вывести на экран значения F , Y для соответствующих значений x

```
PROGRAM Lab2_1;
  Uses Crt;
```

```

                                {Раздел описания данных}
Const
  a=2;
Var
  b, x, Y, F: real;
                                {Раздел операторов – начало алгоритма программы}
BEGIN
  Clrscr;                        {встроенная процедура очистки экрана}
  b:=3.0;
  Writeln('Программа с линейной структурой');
  Writeln('Студент гр.№ Ф.И.О. ');
  Write('Введите значение  $x \geq 0$ '); Readln(x); {ввод значения x}
  Y:=Exp(-a*x)*(x*Sin(a*x+b)-Sqrt(x)*Cos(b*x));
  F:=Ln(Sqrt(Abs(a*x*x-b)))-1;
  Writeln('x=',x:3:2,' Y=',Y:3:1,' F=',F:3:2); {вывод результатов}
  Readln                          {пауза, нажать клавишу ENTER}
END.

```

Результат выполнения программы Lab2_1:

```

Программа с линейной структурой
Студент гр. № Ф.И.О.
Введите значение  $x \geq 0$  3
x=3.00 Y=0.0 F=0.35

```

Задача 2: вычислить сумму двух чисел

```

PROGRAM Lab2_2;
  Uses Crt;
  VAR
    x: integer; y: integer;
    Summa: integer;
BEGIN
  ClrScr;
  Write('Введите первое число: '); Readln(x);
  Write('Введите второе число: '); Readln(y);
  Summa:=x+y;  Writeln('Результат суммирования: ',summa);
  Readln
END.

```

Результат выполнения программы Lab2_2:

Введите первое число: 2

Введите второе число: 4

Результат суммирования: 6

Лабораторная работа № 3

ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ

Цель работы: приобрести навыки работы в системе Turbo Pascal на примере работы с логическими переменными.

Задачи:

- ◆ приобрести практические навыки по организации ввода-вывода логических типов данных;
- ◆ научиться строить алгоритм и писать программы линейной структуры с использованием логических переменных и операций над этим типом данных.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Логический тип данных. Логические выражения и операции

Логический тип данных объявляется следующим образом:
<Идентификатор>:BOOLEAN;

Логические выражения строятся из логических констант и переменных, операций отношения и логических операций. В операциях могут участвовать арифметические и логические выражения, а также символьные данные. Результатом логических выражений является значение True (истина) или False (ложь).

Операции отношения предназначены для сравнения двух величин, результатом является логический тип.

Логические операции:

NOT – НЕ	→ логическое отрицание
AND – И	→ логическое умножение
OR – ИЛИ	→ логическое сложение
XOR – исключающее ИЛИ	→ логическое сложение.

При составлении сложных арифметических или логических выражений необходимо помнить об очередности выполнения опера-

ций того или иного типа. В логических операциях действия выполняются слева направо с соблюдением следующего старшинства:

- 1) NOT;
- 2) *, /, DIV, MOD, AND;
- 3) +, -, OR, XOR;
- 4) операции отношения.

Пример:

```
done:=(5<0);           {done = false}
done:=(c<=2);         {done = true при c<=2}
done:=(c<=10) and (c>=0); {done = true при 0>=c>=10}
```

Линейные алгоритмы и программы

Задача 3: записать условие, которое является истинным, когда: каждое из чисел X и Y нечетное; только одно из чисел X и Y меньше 20.

1. Задать значения аргументов: x, y с помощью оператора ввода (*read*);
2. Записать логические выражения с помощью оператора присваивания, исходя из условия задачи, результатом которых является значение истина.
3. Вывести на экран результаты в виде сообщений и значений логических выражений

Лабораторная работа № 3

Выполнила студентка гр. 110115 Русая А.В.

Тема: "Работа с логическими переменными"

Ввести значения x, y : 13 15

Значения логических переменных: $a=TRUE$ $b=TRUE$

```
PROGRAM Lab3;
```

```
Uses Crt;
```

```
VAR
```

```
  x, y:integer;
```

```
  a, b:boolean; ch:char;
```

```
BEGIN
```

```
  ClrScr;
```

```
  writeln('Лабораторная работа № 3');
```

```
writeln('Выполнила студентка гр. 110115 Русая А.В. ');
writeln;
writeln('Тема: Работа с логическими переменными');
writeln;
write('Ввести значения x, y: '); read(x,y); writeln;
{станд. функция ODD(X) определяет нечетность или четность
  значения параметра X}
b:=odd(x) and odd(y);
a:=((x<20) and (y>20)) or ((x>20) and (y<20));
writeln('Значения логических переменных: a=', a, ' b=',b);
ch:=readkey; {пауза для просмотра результата и выхода из MS
DOS в среду TP 7.0 нажать кл. [Enter]}
```

END.

Лабораторная работа № 4

АЛГОРИТМ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Цель работы: приобрести навыки работы в системе Turbo Pascal на примере программирования нелинейных разветвляющихся алгоритмов.

Задачи:

- ◆ приобрести практические навыки работы с условным оператором (*IF*) и оператором выбора (*CASE*);
- ◆ научиться строить алгоритм и писать программы разветвляющейся структуры.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Разветвляющийся алгоритм это алгоритм, в зависимости от выполнения определённых условий которого, реализуется одно из нескольких, заранее предусмотренных направлений.

Оператор условного перехода (*IF*)

Он дает возможность, в зависимости от заданного в нём условия, выполнить то или иное действие, что позволяет разветвлять вычислительный процесс:

Имеет две отличающиеся конструкции:

- 1) IF <логическое выражение> Then <оператор 1>;
- 2) IF <логическое выражение> Then <оператор 1>
Else <оператор 2>;

Здесь IF (если), Then (тогда), Else (иначе) – ключевые слова. Перед Else ";" не ставится. Оператор 1, 2 – любой оператор языка ТР 7.0, как простой, так и составной.

Работа оператора

Если значение логического выражения "истина", то выполняется оператор 1, а затем оператор, следующий за IF. Если значение логического выражения "ложь", то выполняется оператор, следующий за IF в первой конструкции или оператор, следующий за Else во второй конструкции.

В качестве оператор 1, 2 могут использоваться другие операторы IF.

Пример:

Записать оператор IF для следующей алгебраической схемы:

$$x = \begin{cases} 5 & \text{при } a = b \text{ и } c < d \\ 10 & \text{при } a = b \text{ и } c \geq d \\ 15 & \text{при } a \neq d \end{cases}$$

```
If a=b Then
  If c<d Then x:=5
  Else x:=10
  Else x:=15;
```

В данном примере ";" ставится в конце оператора. Этот же оператор можно записать с использованием трех операторов IF:

```
If (a=b) and (c<d) Then x:=5;
If (a=b) and (c>=d) Then x:=10;
If (a=b) Then x:=15;
```

В примере наличие скобок в логических выражениях операторов IF является обязательным, т.к. операции сравнения имеют более низкий приоритет, чем логические операции.

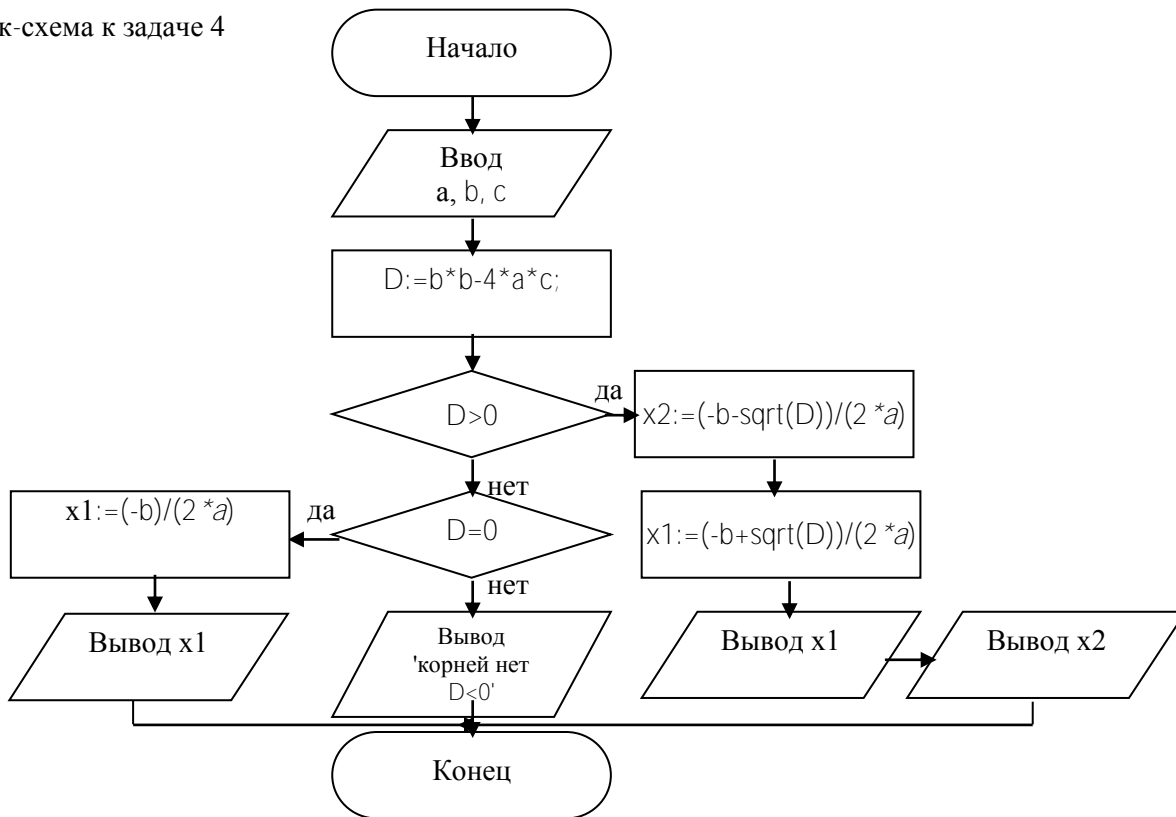
Задача 4: решить квадратное уравнение $ax^2 + bx + c$, где a, b, c – коэффициенты, x_1 и x_2 – корни квадратного уравнения, см. блок-схема к задаче 4.

```
PROGRAM Lab4_1;
  Uses Crt;
  Var
    a,b,c: real; x1,x2: real; D: real;
BEGIN
  ClrScr;
  Writeln('Решение квадратного уравнения');
  Write('Введите коэффициент a: '); Readln(a);
  Write('Введите коэффициент b: '); Readln(b);
  Write('Введите коэффициент c: '); Readln(c);
  D:=b*b-4*a*c;
  if D>0 then
    begin
      x1:=(-b+sqrt(D))/(2*a); x2:=(-b-sqrt(D))/(2*a);
      Writeln('Первый корень уравнения: ',x1);
      Writeln('Второй корень уравнения: ',x2);
    end
  else
    if D=0 then
      begin
        x1:=-b/(2*a);
        Writeln('Уравнение имеет один корень: ',x1);
      end
    else if D<0 then writeln('Корней нет, D<0 ');
  Readln;
END.
```

Результаты выполнения программы Lab4_1:

```
Решение квадратного уравнения
Введите коэффициент a: 4
Введите коэффициент b: 2
Введите коэффициент c: -2
Первый корень уравнения: 0.50
Второй корень уравнения: -1.00
```


Блок-схема к задаче 4



Задача 5: найти $\min\{\max(a,b), \min(c,d)\}$

```
PROGRAM Lab4_2;
  Uses Crt;
  var
    a, b, c, d:integer;
    min, max_ab, min_cd:integer;
BEGIN
  ClrScr;
  writeln('Ввести значения переменных a,b,c,d ');
  readln(a,b,c,d);
  if a>b then max_ab:=a else max_ab:=b;
  if c<d then min_cd:=c else min_cd:=d;
  If max_ab<min_cd then min:=max_ab else min:=min_cd;
  write('максимальное значение из значений ');
  writeln(' а и b=',max_ab);
  write('минимальное значение из значений');
  writeln(' с и d=',min_cd);
  write('минимальное значение из значений max_ab и ');
  writeln('min_cd=',min);
  Readln
END.
```

Результаты выполнения программы Lab4_2:

введите целые значения переменных a,b,c,d

3

8

-2

3

максимальное значение из а и b=8

минимальное значение из с и d=-2

минимальное значение из max_ab min_cd=-2

Оператор выбора (CASE)

Является обобщением оператора IF и позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого селектором, и списка параметров, каждому из которых предшествует список констант выбора. Как и в операторе IF, здесь может присутствовать ELSE, имеющее тот же смысл.

```

CASE <Выражение -селектор> OF
    <Список 1>:<оператор1>;
    ...
    <Список n>:<оператор n>;
else >:<оператор>
END;

```

Работа оператора. Сначала вычисляется значение выражения-селектора, затем выполняется оператор, константа выбора которого равна значению селектора. Если ни одна из констант не равна значению селектора, то выполняется оператор, следующий за Else. Если Else отсутствует, то выполняется оператор, следующий за END. Селектор должен относиться к целочисленному, булевскому, символьному или пользовательскому типу.

Вещественные и строковые типы в качестве селектора не допускаются.

Пример:

```

CASE k+2 OF
    7: A:=k+10;
    8, 9: A:=k+20;
    10..100: A:=k+30
    ELSE A:=10
END;

```

Задача 6: вычислить функцию $F(x)$

$$F(x) = \begin{cases} \frac{x^2}{4} - |x - 2| & \text{если } -1 \leq x \leq 3 \\ \left| \sqrt[3]{2,5|x|} \right| & \text{иначе} \end{cases}$$

```

PROGRAM Lab4_3;
Uses Crt;
Var x:integer; F:real;
BEGIN
    ClrScr;
    write('Ввести значение x '); readln(x);

```

```

Case x of
  -1 .. 3: F:=sqr(x)/4-abs(x-2)
  Else F:=exp(abs((1/3)*ln(2.5*abs(x)))));
End;
writeln('F=',F:3:2);
Readln

```

END.

Результат работы программы Lab4_3:
 Ввести значение x -3
 F=1.96

Лабораторная работа № 5

АЛГОРИТМ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования циклических алгоритмов.

Задачи:

- ◆ приобрести практические навыки по работе с циклами;
- ◆ научиться строить алгоритм с использованием всех видов циклов и писать программы циклической структуры.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Операторы циклов

Вычислительный процесс называется циклическим, если он содержит многократное повторение одних и тех же действий, которое называют ЦИКЛОМ. В языке ПР 7.0 имеются три вида операторов цикла:

- 1) WHILE – оператор цикла с предварительным условием;
- 2) REPEAT – оператор цикла с последующим условием;
- 3) FOR – оператор цикла с параметром.

Оператор цикла с предусловием

Формат оператора цикла с предусловием: **WHILE < логическое выражение > DO <оператор>;**

Здесь <оператор> - любой выполняемый оператор языка, в том числе и составной. Предварительно проверяется значение логического выражения. Пока оно **ИСТИННО** выполняется оператор тела цикла после (DO). Как только оно становится **ЛОЖНЫМ**, происходит выход из цикла. Если с самого начала значение логического выражения **ЛОЖНО**, то оператор не выполняется ни разу. Если логическое выражение никогда не принимает значение **ЛОЖНО**, то происходит заикливание.

Пример:

Для вычисления B в 9-й степени при $B < 0$, когда $\text{EXP}(9 * \text{LN}(B))$, недопустим, можно использовать следующий цикл:

```
K:=1; P:=-1;
While K<=9 Do
  Begin
    P:=P*B;
    K:=K+1
  End;
```

Задача 7: составить программу вычисления и вывода на печать таблицы значений функции $Z = a * \exp(b * x - c * x * x)$ при $X_k \leq X < X_n$ с шагом dX . Здесь $a = -105$; $b = -3.62e-2$; $c = 1,1$; $X_n = 2,65$; $X_k = 5,55$; $dX = 0,15$.

```
PROGRAM Lab5_1;
  Uses Crt;
  Var
    A,B,C:Real;
    X,Xk,Xn,dX:Real; Z:Real;
BEGIN
  ClrScr;
  Read(A,B,C,Xn,Xk,dX); X:=Xn;
  While X<=Xk Do
    Begin
      Z:=A*EXP(B*X-C*X*X);
      Writeln('X=',X;5:2,' ',5,' Z=',Z:10); X:=X+dX
    End;
  Readln
END.
```

Оператор цикла с постусловием

Формат оператора цикла с постусловием:

```
REPEAT
  <Оператор 1>;
  ...;
  <Оператор N>
UNTIL <логическое выражение>;
```

Выполняются операторы циклической части, проверяется значение логического выражения: если оно **ЛОЖНО**, то вновь выполняются операторы циклической части; если же оно **ИСТИННО**, то цикл заканчивается. Если значение логического выражения **ИСТИННО** с самого начала, то операторы циклической части выполняются один раз. Если же логическое выражение никогда не принимает значение "истинно", то операторы тела цикла выполняются бесконечное число раз, т.е. происходит закливание. Нижняя граница операторов тела цикла четко обозначается словом UNTIL, поэтому нет необходимости заключать эти операторы в операторные скобки **Begin – End**. В то же время наличие операторных скобок не является ошибкой.

Пример:

Для вычисления B в 9-й степени при $B < 0$, когда $EXP(9 * LN(B))$, недопустим, можно использовать следующий цикл:

```
K:=1; P:=-1;
Repeat
  P:=P*B;
  K:=K+1
Until K>9;
```

Задача 8: составить программу вычисления и вывода на печать таблицы значений функции $y = \sin(a * X) * \sqrt{X}$ при $X = 5, 6, \dots, 25$; $a = 15.27e-2$.

```
PROGRAM Lab5_2;
Uses Crt;
Var
  A, Y: Real; X: Byte;
```

```

BEGIN
  ClrScr;
  A:=15.27e-2; X:=5;
  Repeat
    Y:=Sin(A*X)*Sqrt(X);
    Writeln('X=',X,' Y=',Y:10); X:=X+1
  Until X>25;
  Readln
END.

```

Примечание: Операторы тела цикла будут выполняться до тех пор, пока X не станет больше 25.

Оператор цикла с параметром

Оператор цикла с параметром имеет два формата. Первый – для возрастающего параметра цикла от начального значения $start$ до конечного значения $finish$ с шагом изменения параметра цикла, равным единице (a). Второй – для убывающего значения параметра цикла от конечного значения $finish$ до начального значения $start$ с шагом, равным минус единице (b).

a) FOR $i:=start$ TO $finish$ DO <оператор>;

b) FOR $i:=start$ DOWNTO $finish$ DO <оператор>;

где **FOR** означает "для", TO, DOWNTO – "до", DO – "выполнить". Организует выполнение одного оператора заранее известное количество раз: где i – параметр цикла; $start$ – начальное значение параметра; $finish$ – конечное значение параметра; <оператор> – простой или составной оператор. Тип переменной цикла i и значений $start$ и $finish$ должен быть порядковым (т.е. целым, символьным, булевским, интервального или перечислимого, за исключением типа REAL)! Шаг изменения параметра цикла всегда равен единице для возрастающих значений параметра (вид – a) и минус единице для убывающих значений параметра цикла (вид – b).

Пример:

Для вычисления B в 9-й степени при $B<0$, когда $EXP(9*LN(B))$, недопустим, можно использовать следующий цикл: $P:=-1$; FOR $K:=9$ DOWNTO 1 DO $P:=P*3$;

Задача 9: составить программу вычисления и вывода на печать таблицы значений функции $y = \text{Sin}(a \cdot X) \cdot \text{Sqrt}(X)$ при $X=5,6,\dots,25$; $a=15.27e-2$.

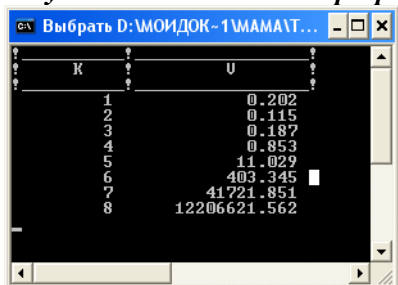
```
PROGRAM Lab5_3;
  Uses Crt;
  Var A,Y:Real; X:Byte;
BEGIN
  ClrScr;
  A:=15.27e-2;
  FOR x:=5 To 25 Do
    Begin
      Y:=Sin(A*X)*Sqrt(X);
      Writeln('X=',X,' Y=',Y:10);
    End; Readln
END.
```

Задача 10: составить программу вычисления произведения

$$V = \prod_{k=1}^m \frac{m^{(k/2)}}{2 \cdot (m-1)} \text{ при } m=8.$$

```
PROGRAM Lab5_4;
  Uses Crt;
  Var K,m:Byte; V:Real;
BEGIN
  ClrScr;
  Writeln('!_____!');
  Writeln('!   K   !   V   !');
  Writeln('!_____!');
  M:=8; V:=1.0;
  For K:=1 To M Do
    Begin
      V:=V*Exp(K/2*Ln(M))/(2*(M-1));
      Writeln(K:10,V:12:3)
    End;
  Readln
END.
```


Результат выполнения программы:



K	U
1	0.202
2	0.115
3	0.187
4	0.853
5	11.029
6	403.345
7	41721.851
8	12206621.562

Задача 11: составить программу вычисления суммы:

$$S = \sum_{i=1}^n \frac{\sin(a \cdot i)}{i}, \text{ где } a=0.1; n=10.$$

```
PROGRAM Lab5_5;  
  Uses CRT;  
  Var K,N:Byte; A,S:Real;  
BEGIN  
  ClrScr;  
  Readln(N,A); S:=0;  
  For K:=1 To N Do  
    Begin  
      S:=S+Sin(A*K)/K; riteln(K:5,S:15:3)  
    End;  
  Readln  
END.
```

Замечания:

1) После прекращения выполнения оператора, значение параметра цикла не определено, за исключением случаев, когда выход из оператора был осуществлён с помощью **GOTO** или стандартной процедуры **Break**.

2) Внутри цикла нельзя изменять ни начальное, ни конечное значения параметра цикла.

3) Если в возрастающем цикле начальное значение больше конечного, то цикл не выполняется ни разу. Для убывающего цикла –

конечное значение не может быть меньше начального. Иначе также не выполняется ни разу.

4) Во всех трех операторах цикла можно использовать операторы условного перехода. Но нельзя передавать управление извне внутрь цикла.

Процедура Break может использоваться во всех циклических операторах. Эта процедура позволяет досрочно выйти из цикла, не дожидаясь выполнения условия выхода.

Процедура Continue может использоваться во всех циклических операторах. Она позволяет перейти к "началу" цикла, или к следующему выполнению тела цикла, даже если выполнение тела цикла не завершено, т.е. она позволяет прекратить выполнение тела цикла. При этом выполнение самого циклического оператора не прекращается.

Функция Random (Range:Word) возвращает псевдослучайное число. Если параметр Range опущен, функция возвращает вещественное число в диапазоне от 0 до 1, иначе – целое число в диапазоне от 0 до Range-1.

Процедура Randomize Инициализирует случайным значением (текущим системным временем) встроенный генератор псевдослучайных чисел.

Пример: вычислить сумму натурального ряда чисел от 3 до 6 (т.е. найти сумму чисел $3+4+5+6$) с помощью трех операторов цикла (For, While, Repeat)

Составим алгоритм задачи:

1) Сначала необходимо определить все данные, которые участвуют в вычислении – это исходные переменные: i – параметр цикла (т.е. количество слагаемых), k – начальное значение равное 3, n – конечное значение равное 6, шаг равный единице и переменная S – вычисляемая сумма (все целые числа).

2) Из условия видно, что для нахождения суммы S , необходимо выполнить i раз суммирование в цикле (т.е. $i = n - k + 1 = 4$), в котором нужно прибавить слагаемое i , причем каждый раз значение i увеличивается на шаг равный единице. Для этого перед началом цикла, необходимо в переменную S послать значение 0, это выполняется с помощью оператора присваивания, $S:=0$; т.к. при выполнении первого шага цикла прибавления первого слагаемого к нулю ($0+3=3$) значение остается равным этому

первому слагаемому. Далее к полученному значению прибавить 4 и т.д., (3+4=7) т.е. выполнить оператор присваивания $S:=S + i$; в цикле.

3) Во время цикла мы также выведем на экран монитора значения переменных i и S в виде таблицы, приведенной ниже:

<u>i</u>	<u>S</u>
3	3
4	7
5	12
6	18

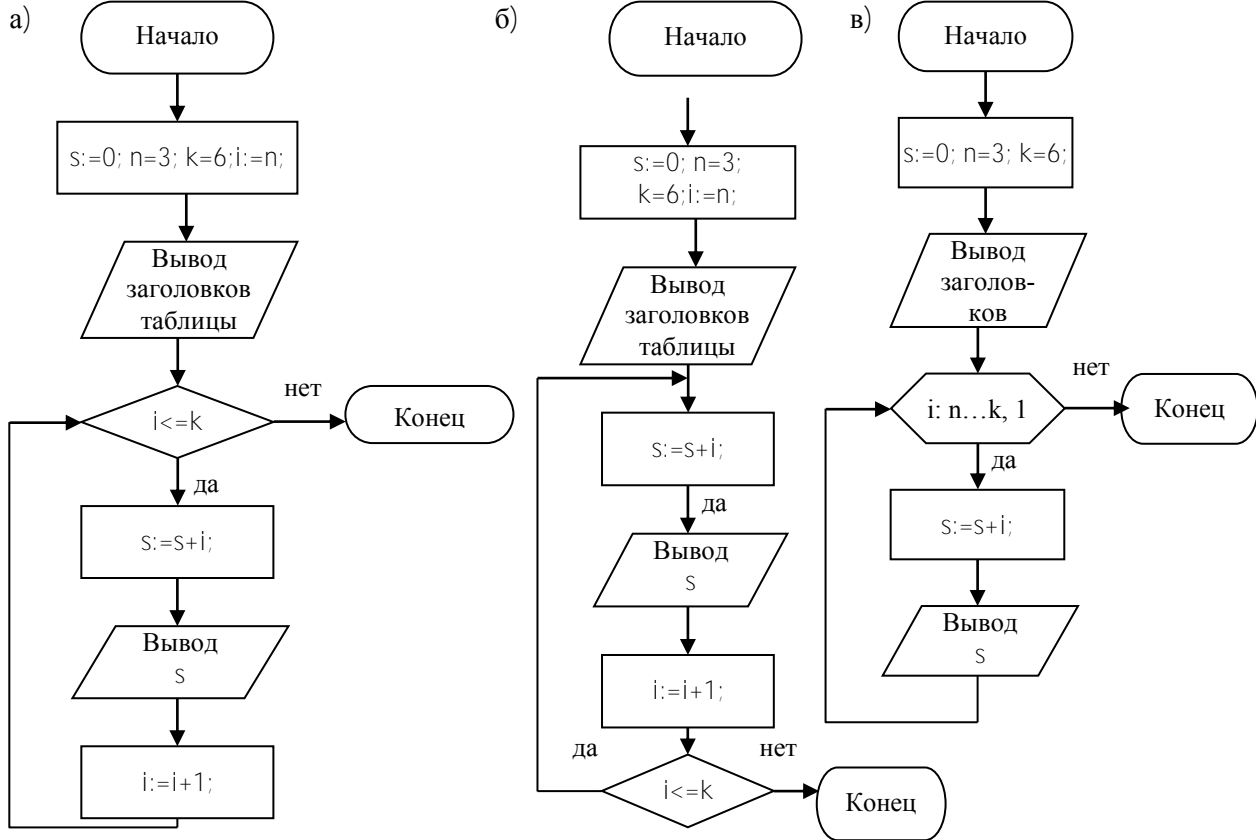
т.к. i , S – целые числа, то в операторе вывода необходимо указать число знаков от первого "!" до второго "!" для переменной i (это число 12), а для переменной S – от второго "!" до третьего "!" (это число 13), тогда оператор вывода значений i , S пишется: **writeln(i:12, s:13);**

4) Цикл завершён при i , свыше конечного значения, затем выход из программы.

Ниже представлены блок-схемы и тексты программ для решения поставленной задачи. Используются все операторы цикла: а) оператор цикла While; б) оператор цикла Repeat; в) оператор цикла For.

Задача 12: вычислить сумму двух чисел, сгенерированных случайным образом.

```
PROGRAM LAB5_9;
  Var x, y:integer; sum:real;t:byte;
BEGIN
  Randomize;
  Repeat
    x:=random(1000); y:=400-random(1000); sum:=x+y;
    Write('Сумма чисел x=',x,');
    Writeln(' и y=',y,' равна ', sum:4:2);
    Write('Завершить программу? 1-да: ');Readln(t);
  Until T=1; Readln
END.
```



a)

```
PROGRAM Lab5_6;
  Var
    i, n, k, s:integer;
BEGIN
  s:=0; n:=3; i:=n; k:=6;
  Writeln('_____');
  Writeln('! i ! s !');
  Writeln('_____');
  WHILE i <=k do
    begin
      s:=s+i;
      Writeln(i:12,s:13);
      i:=i+1
    end
  END.
```

б)

```
PROGRAM Lab5_7;
  Var
    i, n, k, s:integer;
BEGIN
  s:=0; n:=3; i:=n; k:=6;
  Writeln('_____');
  Writeln('! i ! s !');
  Writeln('_____');
  REPEAT
    s:=s+i;
    Writeln(i:12,s:13);
    i:=i+1
  UNTIL i>k;
  END.
```

в)

```
PROGRAM Lab5_8;
  VAR
    i, n, k, s:integer;
BEGIN
  s:=0; n:=3; k:=6;
  Writeln('_____');
  Writeln('! i ! s !');
  Writeln('_____');
  FOR i:=n to k do
    Begin
      s:=s + i;
      Writeln(i:12,s:13)
    End;
  END.
```

Лабораторная работа № 6

АЛГОРИТМ ЦИКЛ В ЦИКЛЕ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования циклических алгоритмов.

Задачи:

- ◆ приобрести практические навыки по программированию цикла в цикле;
- ◆ научиться строить алгоритм с использованием всех видов циклов и писать программы циклической структуры.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Иногда приходится решать задачи, в которых параметр цикла принимает целочисленные значения, но шаг **не равен единице**. Такие задачи можно решить любым оператором цикла, в том числе и For.

Для решения задач с помощью цикла For, при шаге **не равным единице**, необходимо выполнить следующие действия:

а) разделить начальное и конечное значение параметра на шаг, используя функцию round (округление числа до ближайшего целого – результатом является целое число)– эти действия всегда выполняются перед циклом;

б) при вычислениях или выводе значений этого параметра, нужно домножить значение этого параметра на шаг.

Задача 13: вычислить значения функции $Y=X+2$, если X меняет значения от -2 до 4 с шагом равным 2.

```
PROGRAM LAB6_1;
  Uses Crt;
  Cosnt
    h=2;
  Var
    X, Y, Xn, Xk: integer;
BEGIN
  ClrScr;
  Writeln('_____');
  Writeln('! X ! Y !');
```

```

Writeln('_____');
Xn:=-2;           {Xn=-2 – начальное значение X}
Xk:=4;           {Xk =4 – конечное значение X}
Xn:=Round(Xn/h); Xk:= Round(Xk/h); X:=Xn;
FOR X:=Xn to Xk do
  Begin
    Y:=X*h + 2;
    Writeln(X*h:12,Y:13)
  End;
Readln
END.

```

Результат выполнения программы:

X	Y
-2	0
0	2
2	4
4	6

Организация цикла в цикле

Иногда по условию задачи при вычислении функции участвуют два независимых параметра. Рассмотрим решение задачи, когда значения одного параметра не зависят от значений другого.

Для решения таких задач строятся два цикла (внешний и внутренний). Сначала для одного из параметров строится первый цикл – он называется внешний. Внутри первого цикла для каждого из значений строится второй цикл, где изменяются значения второго параметра, это внутренний цикл.

Задача 14: вычислить значение функции $u = \frac{Q+0,2h}{Q+h} - 1$, Q изменяется от 200 до 250 с шагом $\Delta Q=50$, а h изменяется от 0 до 2 с шагом $\Delta h=(h_n-h_k)/2$.

```

PROGRAM Lab6_2;
Const
  dQ=50; hN=0; hK=2;

```

```

var
  Q,QH,QK,i: Integer; h,dh,u:real;
BEGIN
  writeln('_____');
  writeln('|  Q  |  h  |  u  |');
  writeln('-----');
  QH:=200;QK:=250; QH:=round(QH/dQ);
  QK:=round(QK/dQ); dh:=(hK-hH)/4;
  For:= QH to QK Do {внешний цикл по параметру Q}
    Begin
      h:=hH; {параметр внутреннего цикла h меняет значения от
              начального hH до hK с шагом dh для каждого из па-
              раметров Q внешнего цикла}
      While h<=hK do {внутренний цикл по переменной h}
        Begin
          u:=(Q*dQ+0.2*h)/(Q+h)-1;
          writeln(Q*dQ:7,h:9:1,u:11:3);
          h:=h+dh; {изменение значения переменной h на
                   шаг dh внутри внутреннего цикла }
        end; {while}
      end;{for}
    Readln
  END.

```

Результат выполнения программы

Q	h	u
200	0	0.000
200	1.0	0.996
200	2.0	0.992
250	0	0.000
250	1.0	0.997
250	2.0	0.994

Примечание: Существует второй способ решения задачи, когда за внешний принимается цикл по переменной h.

Лабораторная работа № 7

ПОДПРОГРАММЫ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования циклических алгоритмов с применением подпрограмм.

Задачи:

- ◆ приобрести практические навыки по работе с подпрограммами;
- ◆ научиться строить алгоритм с использованием всех видов циклов и писать программы циклической структуры с применением подпрограмм.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Понятие подпрограммы

Подпрограммой называется именованная логически законченная группа операторов, которую можно многократно вызвать для выполнения по имени из различных мест программы. Подпрограммы по структуре сходны с программой, но они обязательно имеют оригинальное имя, которое указывается в заголовке. Подпрограммы описываются в разделе описаний основной программы (глобальный блок).

Для организации подпрограмм используются процедуры и функции.

Подпрограмма-процедура

Описание процедуры имеет вид:

Procedure имя (*формальные параметры*);

Раздел описаний

Begin

Раздел операторов

End;

Здесь **ИМЯ** – имя процедуры. Раздел описаний включает те же разделы, что и в основной программе: **Label**, **Const**, **Type**, **Var** и раздел процедур и функций.

Формальные параметры представляют собой список переменных с указанием их типа. Формальные параметры вместе с круглыми скобками могут отсутствовать. Все типы, используемые в заголовке процедур, должны быть описаны в разделе **Type** основной программы.

Формальные параметры подразделяются на три вида:

- 1) параметры-значения (входные параметры);
- 2) параметры-переменные (выходные параметры);
- 3) параметры процедурного типа.

Описание входных параметров имеет вид:

список переменных 1:тип 1; список переменных 2:тип 2;

Описание выходных параметров соответственно:

Var список переменных 1: тип1; **Var** список переменных 2: тип2;

Вызов процедуры производится в основной программе и имеет следующий вид:

Имя процедуры (фактические параметры);

Фактические параметры представляют собой список параметров, перечисленных через запятую (без указания их типа). Они могут отсутствовать вместе со скобками, в том случае, если отсутствуют формальные параметры. Входными фактическими параметрами, т.е. теми, которые передаются в процедуру, могут быть константы, переменные и выражения. Выходными параметрами (которые получают значения из процедуры) могут быть только переменные.

Между формальными и фактическими параметрами должно быть соответствие по количеству параметров, порядку их следования и типу данных. Имена формальных и фактических параметров могут быть разными или одинаковыми.

Для досрочного выхода из подпрограммы нельзя использовать оператор безусловного перехода **GOTO**. Нельзя перейти по оператору **GOTO** из одной процедуры в другую. С этой целью используется процедура **EXIT**.

Задача 14: подпрограмма-процедура для вычисления суммы квадратов натуральных чисел от 1 до n.

```
PROCEDURE Summa (N:Integer; Var Sum:Integer);  
  Var i:integer;  
BEGIN  
  Sum:=0;  
  For i:=1 To N do Sum:=Sum+sqr(i);  
END;
```

Вызов процедуры в основной программе имеет вид

```
Summa(10,s);
```

Здесь s – переменная типа Integer.

Задача 15: преобразовать угол из градусной меры в радианную меру с помощью подпрограммы-процедуры.

```
PROGRAM Lab7_2;
```

```
  Const  
    pi=3.14;  
  Var  
    x, r:real; ch:char;
```

{описание подпрограммы-процедуры Rad, которая осуществляет перевод градусной меры угла в радианную}

```
  PROCEDURE Rad(Var alfa, betta:real);  
  BEGIN  
    betta:=pi*alfa/180;  
  END;
```

{главная программа}

```
BEGIN
```

```
  Repeat  
    Write('Преобразования угла из градусной меры');  
    Writeln(' в радианную');  
    Write('Введите угол в градусах: '); Readln(x);  
    Rad(x, r); {вызов подпрограммы}  
    Writeln('Ему равен угол в радианах = ',r:6:4);  
    Write('Продолжить вычисления? Enter – да,');
```

```

WriteLn(' Пробел - нет');
ch:=ReadKey;
Until ch=#13;                                {#13 – код символа "пробел"}
END.

```

Результат выполнения программы:

```

Преобразования угла из градусной меры в радианную
Введите угол в градусах: 56
Ему равен угол в радианах = 0.9769
Продолжить вычисления? Enter - да, Пробел - нет
Преобразования угла из градусной меры в радианную
Введите угол в градусах: 67
Ему равен угол в радианах = 1.1688
Продолжить вычисления? Enter - да, Пробел – нет

```

Подпрограмма-функция

Подпрограмма-функция аналогична процедуре, но имеет следующие отличия.

1. Заголовок функции имеет вид:

```

Function имя (список формальных параметров):тип
                результата функции;

```

2. Функция имеет только один результат выполнения.

3. Результат обозначается именем функции, поэтому в разделе операторов функции обязательно присутствовать оператор присваивания, в левой части которого стоит имя функции.

4. Вызов функции в основной программе осуществляется непосредственно внутри выражения по ее имени с указанием фактических параметров.

Задача 16: подпрограмма-функция для вычисления суммы квадратов натуральных чисел от 1 до n.

```

FUNCTION Summa (N:Integer):Integer;
  Var s,i:integer;
BEGIN
  Sum:=0;
  For i:=1 To N do S:=S+sqr(i);
  Summa:=s;
END;

```

Вызов функции в основной программе может иметь вид

Summ:=Summa(10);

Здесь Summ – переменная типа Integer, описанная в главной программе.

Замечание: При использовании подпрограмм процедур и функций следует иметь в виду, что переменные, представленные в разделе описаний основной программы (Program), действуют в разделе операторов основной программы и в любой ее подпрограмме. Эти переменные называются **глобальными**. Переменные, описанные в подпрограмме, действуют только в этой подпрограмме и в любой в ней процедуре и функции. Такие переменные называются **локальными**. Они недоступны для операторов основной программы и других подпрограмм.

Рекурсия. Рекурсивный алгоритм. Рекурсией называется ситуация, когда какая-то подпрограмма прямо или через другие подпрограммы вызывает себя в качестве подпрограммы. Реализуемый при этом алгоритм называется рекурсивным.

Рекуррентные соотношения. В математике очень часто встречаются последовательности чисел, в которых каждый следующий член выражается через предыдущие члены. В арифметической прогрессии, например, каждый следующий член равен предыдущему, увеличенному на разность прогрессии: $a\{i\}=a\{i-1\}+d$.

Формулы, выражающие очередной член последовательности через один или несколько предыдущих членов, называют **рекуррентными соотношениями**.

Задача 17: вычислить $n!$. ($n!=1*2*3*...*n$)

```
PROGRAM Lab7_2;
  Var
    n:integer;
    {описание подпрограммы-функции}
  FUNCTION Fact(i:integer):longint;
  Begin
    if i=0 then Fact:=1 else Fact:=i*Fact(i-1);
  End;
```

```

                                {главная программа}
BEGIN
  ClrScr;
  write('Введите число n: '); readln(n);
  writeln('Факториал n=',Fact(n));      {вызов функции Fact}
END.

```

Задача 18 вычислить $n!$ с использованием прямой рекурсии для вычисления $n!$.

```

PROGRAM Lab7_3;
  Uses Crt;
  Var n:integer;

                                {Рекурсивная функция}

FUNCTION Fact(n:integer):real;
Begin
  If n=0 then Fact:=1
    else Fact:=n*Fact(n-1);
End;

```

```

BEGIN
  ClrScr;
  Repeat
    Writeln(' Введите положительное n<=33');Readln(n);
    Writeln(Fact(n));
  Until Eof;
  Readln
END.

```

Примечание: для выхода из этой программы можно задать значение $n > 33$ или нажать *Ctrl/z* и *Enter*. При $n > 33$ возникает переполнение при умножении чисел с плавающей запятой.

Лабораторная работа № 8

ФАЙЛЫ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования циклических алгоритмов с применением файлов.

Задачи:

- ◆ приобрести практические навыки по работе с файлами;
- ◆ учиться строить алгоритм с использованием всех видов циклов и писать программы циклической структуры, записывать входные и выходные данные в файл любого типа.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Общие сведения о файлах

Для хранения информации полученной в результате работы программ, используют различные внешние запоминающие устройства: гибкие магнитные диски, винчестера, компакт-диски и другие носители. Использование внешних запоминающих устройств для хранения информации является наиболее надёжным и удобным способом хранения информации.

Физический файл

Файл – это поименованная область памяти на внешнем носителе, предназначенная для хранения информации. В таком понимании файл называют физическим файлом. Файл может быть связан с любым источником или потребителем информации: клавиатурой, принтером, магнитным диском и др. Мы будем рассматривать только стандартные файлы языка Pascal и дисковые файлы, созданные пользователем, как имеющие самое широкое применение при работе ПК

Логический файл

Для организации работы с физическими файлами в языках программирования предусмотрены специальные структуры данных - тип данных файл. Файл в таком понимании называют логическим файлом, так как в этом случае он представляет собой лишь логическую модель хранения информации не зависящую от организации конкретного физического файла.

Тип данных - файл

Файл - это структурированный тип данных, состоящий из последовательности компонентов (элементов) одного типа и одной длины. Количество компонент в файле заранее не оговаривается. Объявляется тип данных файл в разделе объявления типов с помощью служебного слова FILE, после которого указывается тип компонент файла (базовый тип). Базовым типом может быть любой тип, кроме типа FILE.

Type

FV = FILE of integer;

Var as:FV;

Переменные типа FV будут файлами с целочисленными компонентами. В таком случае переменную FV можно применять как параметр процедур или функций. Можно также объявить файловую переменную непосредственно в разделе объявления переменных, но тогда подпрограммы для неё недоступны:

Var

as: file of byte;

Работа с файлами

Для простоты и наглядности файловую переменную можно изобразить как последовательность однотипных компонент, хранящихся на винчестере. В самом начале записывается имя файла, после него компоненты файла, а в самом конце признак конца файла. Считывание или запись информации из файла осуществляется с помощью воображаемой магнитной головки, которую мы в дальнейшем будем называть текущим указателем.

Файл может находиться в различных состояниях:

1. Файл закрыт;
2. Файл открыт для записи;
3. Файл открыт для чтения.

Установочные и завершающие операции

➤ **Assign(var FV; name: string);** – процедура Assign предназначена для связывания файловой переменной (*логического файла*) с конкретным файлом на диске (*физическим файлом*), где FV - имя файловой переменной (идентификатор); name - строка содержащая имя физического файла в операционной системе MS-DOS.

Имя файла может содержать путь к файлу, например: D:\TP\stud.dat. При отсутствии пути к имени файла подразумевается файл, находящийся в текущем каталоге. Так, например, процедура Assign(ff,'bank.dat') связывает файловую переменную (логический файл) ff с физическим файлом на диске bank.dat. После выполнения этой процедуры файловая переменная ff будет представителем в программе физического файла на диске bank.dat.

➤ **Reset(var FV);** – процедура Reset предназначена для подготовки к чтению или к записи существующего физического файла, связанного с файловой переменной FV. Файл должен обязательно существовать. Если файл не пуст, текущий указатель перемещается к первой компоненте файла. В противном случае текущий указатель перемещается к концу файла. Файл становится открытым для чтения и записи.

➤ **Rewrite(var FV);** – процедура Rewrite предназначена для создания нового файла, или для подготовки к записи с начала существующего файла (к перезаписи). Текущий указатель перемещается к началу файла. Если файл существует, то его предыдущее содержимое уничтожается. Если файла нет, то он будет создан. Файл становится открытым для записи и закрытым для чтения.

➤ **Close(var FV);** – закрывает открытый файл. Файл становится закрытым для чтения и закрытым для записи. Ни в коем случае не следует пренебрегать этой операцией. Оставшийся открытым файл, как правило, может вызвать аварийную ситуацию.

Операции ввода-вывода

➤ **Read(FV,v1[,v2,...,vn]);** – процедура Read предназначена для чтения значений из файла в программу. FV – файловая переменная, **v1[,v2,...,vn];** – переменные (квадратные скобки обозначают не обязательность присутствия тех переменных, которые в них указаны), в которые будет помещаться информация из компонент файла. Тип этих переменных должен совпадать с типом компонент переменной VF в файле. При прочтении компоненты из файла текущий указатель перемещается к следующей компоненте.

➤ **Write((FV,v1[,v2,...,vn]);** – процедура Write предназначена для записи информации в файл. FV – файловая переменная, **v1[,v2,...,vn];** – переменные того же типа, что и компоненты переменной VF, содержимое которых будет помещено в файл.

Перемещения по файлу

Все компоненты типизированных файлов проиндексированы начиная с нуля, что позволяет организовать произвольное перемещение по файлу и доступ к любой компоненте (элементу) такого файла в любой момент времени.

- **Seek(var FV, n:longint);** – перемещает текущий указатель в файле FV на позицию номер n.
- **Seek(var FV, FileSize(var FV));** – установка указателя за последним компонентом, которая короче и эффективнее, чем поиск конца файла путем перебора всех компонент в файле.
- **Truncate(var FV);** – усекает файл (обрезает его) на текущей позиции и подготавливает файл для записи.
- **FileSize(var FV):longint;** – возвращает текущий размер файла (сообщает о количестве компонент в файле).
- **FilePos(var FV):longint;** – определяет текущий номер компонента, на котором установлен в данный момент указатель.
- **EoF(var FV):boolean;** – функция определения конца файла. Возвращает значение true, если текущий указатель находится за последней компонентой файла, false в противном случае.

Текстовые файлы

Кроме описанных выше файлов в Turbo Pascal 7.0 применяется специальный вид файлов, предназначенный в основном для работы с текстовой информацией. Этот тип файлов объявляется с помощью служебного слова **text**, например

```
Var t: text;
```

Работа с текстовыми файлами немного отличается от работы с обычными файлами. Информация в таких файлах хранится построчно. Компонентами текстовых файлов являются строки различной длины, разделённые между собой специальным признаком конца строки. Компоненты текстовых файлов не индексируются, поэтому произвольный доступ к ним невозможен.

Процедуры и функции для работы с текстовыми файлами

- **Append(var FV: Text);** – открывает существующий текстовый файл для добавления в конец информации.
- **Readln(var FV: Text; V1, [V2, V3,..., Vn]);** – читает информацию из текстового файла. V1, V2,... Vn – переменные целого, вещественного или строкового типа в которые помещается прочитанная информация.
- **Writeln(var FV: Text; V1, [V2, V3,... Vn]);** – записывает

информацию в текстовый файл и вводит в файл признак конца строки. V1, V2,..., Vn - переменные целого, вещественного или строкового типа, содержащие информацию, помещаемую в файл.

Процедуры **Readln** и **Writeln** осуществляют те же действия, что и процедуры **Write** и **Readln**, но после операций чтения и записи производят переход к следующей строке текстового файла. Процедура **Read** переходит к следующей строке только в случае исчерпания текущей строки. Процедура **Readln** позволяет совершить этот переход, не дожидаясь конца строки.

➤ **Eoln(var FV: Text);** – функция определения конца строки. Возвращает значение **true**, если текущий указатель находится за последней компонентой строки, **false** в противном случае.

➤ **SeekEoln(var FV: Text): boolean;** – производит поиск конца текущей строки файла. Перемещает указатель к следующему значению в строке. Если достигнут конец строки возвращается **true**.

➤ **SeekEof(var FV: Text): boolean;** – производит поиск конца файла. Перемещает указатель к следующему значению. Если достигнут конец файла возвращается **true**.

Задача 19: фрагменты программ с применением процедур и функций для работы с файлами.

```
PROGRAM Lab8_1;
  Uses Crt;
  Var
    F:File Of Byte;
    Size:LongInt;

    {Установить указатель на середину ранее созданного файла}

BEGIN
  ClrScr;
  Assign(F, ParamStr(1)); Reset(F);
  Size:=FileSize(F);
  Writeln('Размер файла: ', Size,' байт.');
```

Writeln('Установка указателя позиции на середину файла');

```
  Seek(F, Size Div 2); Writeln('Текущая позиция: ', FilePos(F));
  Close(f); Readln
END.
```

Задача 20: создание нового текстового файла

```
PROGRAM Lab8_2;  
  Uses Crt;  
  Var F:Text;  
BEGIN  
  ClrScr;  
  Assign(F, 'TEST.TXT');  
  ReWrite(F); {Создаем новый файл}  
  WriteLn(F, 'Немножко текста ;-');  
  Append(F); {Добавляем данные к концу файла}  
  WriteLn(F, 'Еще немножко текста !');  
  Close(F); {Закрываем файл}  
  ReadLn  
END.
```

Задача 21: создать текстовый файл с 8-ю цифрами и пробелами в конце строк

```
PROGRAM Lab8_3;  
  Uses Crt;  
  Var F:Text;  
      I, J:Integer;  
BEGIN  
  ClrScr;  
  Assign(F, 'TEST.TXT'); ReWrite(F);  
  WriteLn(F, '1 2 3 4 '); WriteLn(F, '5 6 7 8 ');  
  Reset(F);  
  {Считывает данные из файла Test.Txt. Процедура SeekEoln возвращает значение TRUE, если в файле больше нет текста (кроме пробелов), т.е. если в данной строке больше нет цифр}  
  While Not SeekEof(F) Do  
    Begin  
      If SeekEoln(F) Then ReadLn;  
      Read(F, J); Write(J, ' ');  
    End;  
  ReadLn  
END.
```

Задача 22: создать файл целых чисел из N элементов. Вычислить среднее арифметическое элементов файла

```
PROGRAM Lab8_4;
  Uses Crt;
  Type
    ff=file of integer;
  VAR
    log_f: ff; i: integer; N:integer; element:integer;

  PROCEDURE Zapoln_file_elem(var f: ff);
  Begin
    Write('Введите количество элементов в файле: ');
    Readln(N); Rewrite(f);
    Writeln('Введите значения элементов файла: ');
    for i:=1 to N do
      begin
        Read(element); Write(f, element);
      end;
    Close(f);
  End;

  PROCEDURE Read_elem_iz_file(var f: ff);
  Begin
    Reset(f);
    while not eof(f) do Read(f,element);
    Close(f); Writeln;
  End;

  FUNCTION Sr_ar(var f:ff): real;
    var sum: real;
  Begin
    Reset(f); sum:=0;
    while not eof(f) do
      begin
        Read(f, element);
        sum:=sum+element;
      end;
    sr_ar:=sum/FileSize(f); Close(f);
  End;
```

```

BEGIN
  ClrScr;
  Assign(log_f,'data.dat'); ClrScr;
  Zapoln_file_elem(log_f);
  Writeln('Содержимое файла: ');
  Read_elem_iz_file(log_f);
  Write('Среднее арифметическое элементов файла= ');
  Writeln(' sr_ar(log_f):3:1);
  Repeat Until Keypressed;
END.

```

Задача 23: вычислить сумму и произведение квадратов натуральных чисел от 1 до 4. Записать произведение и сумму в файл целых чисел с именем **Tabliza**. Из данных файла **Tabliza** сформировать текстовый файл с именем **Tabliza1** в виде таблицы. Записать эту таблицу на экран.

```

PROGRAM Lab8_5;
  Const n=4;
         namfT='tabliza1'; namfI='tabliza';
  Type
    ft1=text; fi1=file of integer;
  Var
    i,sum,pr:integer; fi:fi1;ft:ft1;

  FUNCTION Summa:integer;
  Begin
    summa:=sqr(i);
  End;

  PROCEDURE Zap_pr_sum(var pr1,sum1:integer);
  Begin
    for i:=1 to n do
      begin
        sum1:=sum1+summa;
        pr1:=pr1*summa; write(fi,pr1,sum1);
        writeln(ft,i:8,sum1:8,pr1:9);
      end;
  End;

```

```

PROCEDURE Read_zapis_tabl(var pr2, sum2 :integer);
Begin
    for i:=1 to n do
        begin
            read(fi,pr2,sum2);
            writeln(i:8,sum2:8,pr2:9);
        end;
    End;

PROCEDURE Tabl;
Begin
    writeln('-----');
    writeln('| I | SUM | PR |');
    writeln('-----');
End;

PROCEDURE Tabl_file;
Begin
    writeln(ft,'-----');
    writeln(ft,'| I | SUM | PR |');
    writeln(ft,'-----');
End;

BEGIN
    ClrScr;
    assign(ft,namft);
    {assign(ft,'tabliza1');}
    rewrite(ft);
    tabl_file;
    assign(fi,namfi);
    {assign(fi,'tabliza');}
    rewrite(fi); tabl;
    sum:=0; pr:=1;
    append(ft);
    Zap_pr_sum (pr,sum);
    reset(fi); Read_zapis_tabl(pr,sum);
    close(ft); close(fi); Readln;
END.

```

Лабораторная работа № 9

ОДНОМЕРНЫЕ МАССИВЫ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования одномерных массивов.

Задачи: построение алгоритмов с использованием одномерных массивов.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Понятие массива

Массивы относятся к структурированным типам данных в ТР 7.0. Массивы наиболее часто используются при подготовке статистических сводок, справочных изданий, словарей и т.д. Под массивом подразумевают совокупность конечного числа данных одного типа, объединенных одним именем. Доступ к отдельным элементам массива осуществляется с помощью общего имени и порядкового номера (индекса) необходимого элемента. Имена массивов образуются так же, как имена простых переменных.

Возможны два способа описания массивов:

```
Type <имя типа> = Array[i,i,...,i] Of <тип элементов> ;  
Var <имя массива>:<имя типа>;
```

или

```
Var <имя массива>:Array[i,i,...,i] Of <тип элементов>;
```

Здесь [i,i,...,i] – типы индексов массива (любой скалярный тип, кроме типа Real), хотя наиболее часто в качестве индексов элементов массива применяют целые числа. Служебные слова: Array, Of означают соответственно *массив* и *из*. Тип элемента – это переменные любого типа данных, кроме файловых типов. Массивы бывают одномерными и многомерными. Количество индексов определяют размерность массива. Размер массива ограничивается только объемом рабочей памяти ПК.

Объявление одномерного массива:

```
VAR  
<имя массива>:ARRAY[нач_индекс..кон_индекс] OF <тип данных>;
```


Величины, обозначающие "начальный индекс" и "конечный индекс", в квадратных скобках разделяются двумя (!) точками, причём перед этими двумя точками и после них допускаются пробелы.

Пример:

Var Index: array[1..5] of Integer; – объявлен одномерный массив с именем Index из 5 целых чисел;

Var Ball:array[11..20] of real; – объявлен массив Ball; из 10 вещественных чисел, индекс которых меняется от 11 до 20;

Var Name:array[1..5] of string[10]; – объявлен массив Name, элементами которого являются строки, каждая из которых может иметь длину до 10 символов включительно.

В следующем примере объявлены массивы, содержащие не только элементы различных типов, но также и индексы различных простых типов:

Var www:array [(mon,tue,whed)] of integer; – описан массив из трёх целых чисел, индексы элементов массива имеют перечислимый тип и принимают значение названий дней недели mon, tue, whed.

Var ast:array ['A'..'Z'] of boolean; – описан массив элементов логического типа, тип индексов – ограниченный символный.

Var art:array [(black,white)] of 11..20; – описан массив целых чисел с индексами black, white. Каждый элемент массива может принимать значения в диапазоне от 11 до 20.

Var svz: array [byte] of integer; – описан массив из 256 целых чисел с индексами стандартного типа byte (от 0 до 256).

Обратиться к любому элементу массива можно с помощью имени массива и индекса, например Index[5] – обращение к 5-ому элементу массива Index.

Операции над массивами и их совместимость

Операции над элементами массивов определяются типом элементов каждого массива (Real, Integer, Char и т.д.). Причем элементы двух массивов считаются совместимыми, если их типы совместимы.

Однако, совместимость элементов массивов не означает совместимости самих массивов.

Пример:

1) Var

X:array[1..5] of Integer; Y:array[1..5] of Integer;

Массивы X, Y имеют различные типы и полное присваивание X:=Y; для них недопустимо, хотя поэлементное X[1]:=Y[5]; правомерно, поскольку оба элемента имеют целый тип.

2) Массивы C и D, описанные в виде:

Type

Mas=array[1..5] of real;

Var

C:Mas; D:Mas;

или

Var

C,D:array[1..5] of real;

совместимы, поскольку в первом описании с Type имеется ссылка на один и тот же тип, а в следующем описании с Var массивы C и D представлены в одном списке. Над ними допустимо присваивание C:=D;

Извлечения и присвоения в массивах

В отличие от стандартных переменных массивы не могут обрабатываться целиком, обработка возможна только поэлементно¹. Но можно получить доступ к каждому элементу-ячейке массива. Это выполняется путём указания значения индекса в квадратных скобках. Так, например, с помощью оператора mas[2]:=34; элементу массива с индексом 2 присваивается значение 34. Оператор Writeln(mas[2]); вызовет вывод на экран пользователя значение хранящегося в элементе-ячейке N 2 массива mas. наряду с конкретным значением (константой) в качестве индекса может быть использована переменная, например, при обработке массива

¹ Исключение составляют упаковочные массивы символьного типа, которые можно выводить в стандартный файл помассивно (строками символов).

поэлементно в рамках цикла "FOR...TO...DO". Так помощью фрагмента программы

```
FOR i:=1 TO 25 DO mas[i]:=0;
```

всем элементом массива присваивается значение "0". Использование массивов вместо одиночных переменных позволяет, благодаря применению циклов "FOR TO DO" существенно сэкономить время при написании программ.

Начальные значения элементов массива могут быть введены сразу с описанием массива как типизированная константа в разделе CONST. При этом могут применяться две формы описания:

```
Type <имя типа> = Array[<тип индекса>] Of <базовый тип
элементов>;
Const <имя константы>:<имя типа>=(<список констант>);
```

или

```
Const имя константы:Array[тип индекса] Of базовый тип эле-
ментов(список констант);
```

Пример:

```
Type
TX=Array[1..5] Of Real;
Const
X:TX=(2.3,3.5,2.8,3.2,2.0);
```

или

```
Const
X:Array[1..5] Of Real=(2.3,3.5,2.8,3.2,2.0);
```

Разумеется, что значения элементов этих массивов изменять в программе недопустимо.

Ввод и вывод массивов

Для ввода и вывода числовых значений элементов массива используются операторы цикла и операторы ввода-вывода. Например, цикл

```
FOR i:=1 TO 15 DO Read(M[i]);
```

организует ввод 15 значений элементов массива M, а цикл

```
FOR i:=1 TO 15 DO Write(M[i], ' ');
```

вывод этих элементов через пробел, вместо пробела можно использовать форматированный вывод E:m:n – для вещественных чисел и E:m – для целых чисел.

Пример заполнить массив 10-ью случайными числами из диапазона от 0 до 100.

```
For i:=1 to 10 do Massv[i]:=random(100);
```

Задача 24: дана последовательность действительных чисел s_1, \dots, s_5 . Организовать массив для хранения этих чисел. Использовать подпрограммы.

```
PROGRAM lab9_2;
  Uses Crt;
  Const m=5;
  Type mas=array[1..m] of real;
  Var
    i,n,x,r:integer;
    s:mas; ch:char;

    {процедура ввода одномерного массива с клавиатуры}
  PROCEDURE Vvod_mas(var s:mas);
  Begin
    write('ввод значений элементов ');
    writeln('одномерного массива');
    for i:=1 to m do
      begin
        write('введите ',i, ' элемент последовательности ');
        readln(s[i])
      end;
    end;
  end;

  {процедура вывода элементов одномерного массива в строку}
  PROCEDURE Zapis_Mas(var s:mas);
  Begin
    for i:=1 to m do
```

```

        write(s[i]:7:1);
    writeln
end;
BEGIN
    ClrScr;
    Vvod_Mas(s);
    Writeln('Одномерный массив');
    Zapis_Mas(s);
    ch:=readkey
END.

```

Результат выполнения программы:

```

Введите 1 элемент последовательности 3.5
Введите 2 элемент последовательности 2.5
Введите 3 элемент последовательности 0.5
Введите 4 элемент последовательности 1.5
Введите 5 элемент последовательности 3.0
Одномерный массив
3.5 2.5 0.5 1.5 3.0

```

Методы обработки массивов

Массивы – наиболее часто применяемый тип данных при обработке большого количества информации, различного рода статистических данных и т.д. Это объясняется не столько удобством хранения информации в памяти компьютера в виде массива, сколько возможностью легко производить различного рода обработку массивов.

В настоящей лабораторной работе описаны наиболее простые и часто встречающиеся ситуации при обработке информации в массивах. Все они достаточно просты, и поэтому снабжены не подробными комментариями, а конкретными примерами. Рекомендуем Вам самостоятельно разобрать каждый из предложенных примеров. Не переходите к следующему разделу (примеру программы), если Вам не понятно назначение каждой строки предыдущего примера программы

Пример: фрагмента определения суммы положительных элементов массива.

```

PROGRAM lab9_3;
  CONST
    k=10;
  TYPE
    mas=ARRAY[1..k] of real;
  VAR
    m:mas; i: integer; s:real;

  PROCEDURE Sum_Pol(var m:mas);
  Begin
    FOR i:=1 TO k DO if m[i]>=0 then s=s+m[i];
  End;
BEGIN
  ...
  s:=0;
  sum_pol(m);
  ...
END.

```

Пример: фрагмента определения максимального элемента одномерного массива и его порядкового номера.

```

PROGRAM lab9_4;
  CONST
    k=10;
  TYPE
    mas=ARRAY[1..k] of real;
  VAR
    m:mas; max:real;
    i: integer;
    t: integer; {порядковый номер максимального элемента}
  PROCEDURE Max_Elem(var m:mas;var t:integer);
  Begin
    FOR i:=1 TO k DO
      if m[i]>max0 then
        begin
          max:=m[i]; t:=i;
        end
    End;

```

BEGIN

...

t:=1;

max:=m[1];

max_elem(m,t);

Writeln('Максимальный элемент равен ',max:5:2);

Writeln('Номер максимального элемента ',t);

...

END.

Пример: фрагмент поиска элемента в неупорядоченном массиве

PROGRAM lab9_4;

CONST k=10;

TYPE

mas=ARRAY[1..k] of real;

VAR

m:mas; i: integer;

p:real;

{значение искомого элемента}

t integer

{индекс (номер) иском. элемента}

PROCEDURE Poisk_Elem(var m:mas;var t:integer);

Begin

FOR i:=1 TO k DO

begin

...

Write('Введите контрольное число '); Readln(p);

t:=0; {допустим, что в массиве нет такого элемента}

FOR i:=1 TO k DO

if m[i]=p then {проверка нашего утверждения}

begin

t:=i; write('Эл. N ',i,' равен искомому');

end;

if t=0 then write('В массиве нет такого элемента');

End;

BEGIN

...

END.

Сортировка массивов. В большинстве случаев, массивы применяются для хранения большого количества однотипной информации, создания и организации баз и банков данных и т.д., поэтому часто возникает задача об упорядочивании массива.

Рассмотрим простой случай такой задачи: дан числовой массив x_1, x_2, \dots, x_n , элементы которого попарно различны; требуется переставить элементы массива так, чтобы после перестановки они были упорядочены в порядке возрастания: $x_1 < x_2 < \dots < x_n$. Существует несколько алгоритмов решения этой задачи. Мы рассмотрим два наиболее популярных.

Алгоритм сортировки выбором Очевидно, что первое место в массиве должен занять минимальный элемент массива, второе - наименьший из всех остальных, третий - наименьший из оставшихся и т.д.

Для этого необходимо выполнить следующую последовательность действий:

1. Определить минимальный элемент массива;
2. Поменять его местами с первым элементом;
3. Определить минимальный элемент среди оставшихся элементов;
4. Поменять его местами со вторым элементом и т.д.;

Эта последовательность действий должна выполняться до тех пор, пока не будет определён последний минимальный элемент.

Всю операцию по упорядочиванию массива можно разбить на более простые задачи.

Первая - поиск минимального элемента.

Вторая - замена местами минимального элемента с номером t с элементом N_1 , в первом случае, и с номером i в общем случае. Для этого необходимо объявить дополнительную переменную buf , тип которой должен совпадать с типом элементов массива, и выполнить последовательность действий:

```
buf:=m[t]; {сохранить в буфере значение минимального элемента}  
m[t]:=m[i]; {в ячейку, хранившую минимальный элемент записать значение элемента  $N_i$ }  
m[i]:=buf; {в ячейку  $N_i$  записать из буфера значение минимума}
```

Фрагмент программы, который выполняет полную сортировку массива состоящего из K элементов по возрастанию (от меньшего

элемента к большему)

```
FOR i:=1 TO K-1 DO
  BEGIN
    t:=i;    {предположим, что i-й элемент минимален}
            {цикл для поиска мин. в оставшейся части}
    for j:=i+1 to K do
      if m[j]<m[t] then t:=j; {условие для определения минимума}
      buf:=m[t]; m[t]:=m[i]; m[i]:=buf; {замена}
    END;
```

Описанный метод сортировки массивов можно применять как для сортировки массивов по возрастанию, так и для сортировки массивов по убыванию. Для этого просто необходимо определять не минимальный элемент массива, а максимальный. В тексте программы это выражается заменой в некоторых местах знака "<" на знак ">".

Алгоритм пузырьковой сортировки

Алгоритм так называемой "пузырьковой" сортировки более оригинален и в большинстве случаев более эффективен. При использовании этого алгоритма весь массив просматривается несколько раз подряд. При каждом таком просмотре сравниваются последовательно только соседние элементы массива: сначала первый со вторым, потом второй с третьим, и в конце предпоследний с последним. Если при сравнении окажется, что предыдущий элемент больше следующего, они меняются местами описанным выше способом. Так в результате одного последовательного просмотра элементы, значение которых больше, "всплывают" образно говоря на поверхность, т.е. ближе к концу массива. Если провести такой последовательный просмотр массива несколько раз, то "тяжёлые" элементы окончательно "всплывут" и массив окажется отсортированным. Остаётся нерешённым ещё один вопрос. Сколько раз необходимо выполнять такой просмотр? Ведь может оказаться, что и одного просмотра будет достаточно. Столько, сколько нужно для полной сортировки, т.е. пока при просмотре элементы не будут меняться местами. Для этого удобно организовать логическую переменную *ind*, и присваивать ей значение ложь в случае, если замена происходила, хотя бы один раз. Если значением этой переменной останется истина, то просмотры

необходимо прекратить.

Фрагмент программы реализован на основе метода пузырьковой сортировки. Он выполняет полную сортировку массива состоящего из K элементов.

```
REPEAT
  ind:=true;      {предположим, что массив уже отсортирован}
  FOR i:=1 TO K-1 DO    {цикл для организации просмотра}
    if m[i]>m[i+1] then  {сравнение двух соседних элементов}
      begin
        buf:=m[i];      {меняем соседние элементы местами}
        m[i]:=m[i+1]; m[i+1]:=buf;
        ind:=false;    {как оказалось, массив неотсортирован}
      end;
UNTIL ind;        {выполняем просмотры пока ind=false}
```

Лабораторная работа № 10

ФАЙЛЫ И ОДНОМЕРНЫЕ МАССИВЫ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования одномерных массивов с применением файлов типа массив и текстовый.

Задачи: построение алгоритмов с использованием одномерных массивов и файлов типа массивы и текстовые.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Работа с одномерными массивами и файлами

Запись в файл элементов одномерных массивов и чтение элементов массивов из файла производится с помощью циклов. Для записи в файл типа массив необходимо в процедурах WRITE и READ записать файловую переменную типа массив и через запятую имя массива без индекса, в отличие от текстового файла, где обязательно в квадратных скобках нужно указать индекс элемента массива рядом с именем массива.

Пример:

<i>Запись элементов массива в файл типа массив</i>	<i>Запись элементов массива в текстовый файл</i>
<pre>Const n=7; Type mas=array[1..n] of real; f=file of mas; var m1:mas; fm:f; i: integer; Begin assign(fm, 'name_file'); rewrite(fm); for i:=1 to n do begin write('введите ',i, '); writeln(' элемент'); readln(m1[i]); write(fm,m1); end; close(fm); end.</pre>	<pre>Const n=7; Type mas=array[1..n] of real; txt=text; var m2:mas; ft:txt; i: integer; Begin assign(ft, 'name_file'); rewrite(ft); for i:=1 to n do begin write('введите ',i, '); writeln(' элемент'); readln(m1[i]); write(ft,m2[i]:5:1); end; close(ft); end.</pre>
<i>Чтение элементов массива из файла типа массив</i>	<i>Дозапись элементов массива, ≤ 0, в текстовый файл</i>
<pre>Const n=7; Type mas=array[1..n] of real; f=file of mas; var m1:mas; fm:f; i: integer; Begin assign(fm, 'name_file'); reset(fm); for i:=1 to n do read(fm,m1); close(fm); end.</pre>	<pre>Const n=4; Type mas=array[1..n] of real; txt=text; var m2:mas; ft:txt; i: integer; Begin assign(ft, 'name_file'); Append(ft); for i:=1 to n do if m2[i] <=0 then write(ft,m2[i]:5:1); close(ft); end.</pre>

Задача 25: исходный одномерный массив записать в файлы: типа массив и текстовый. Дозаписать в текстовый файл, сформированный одномерный массив, только из положительных элементов.

PROGRAM Lab10;

Uses Crt;

Const n=7;

Type mas=array[1..n] of real;

txt=text;

fil=file of mas; filename=string;

var

i, k: integer; ch:char;

nam_m, nam_t:filename; {nam_m - имя файла типа *mas*,
nam_t – имя файла типа *text*}

f_t1:txt; {файловая переменная типа *text*}

f_m:fil; {файловая переменная типа *mas*}

{процедура записи одномерного массива в файл типа *mas*}
{и в текстовый файл типа *text*}

PROCEDURE Zapis_Mas(nam_m,nam_t:filename;

var m1:mas;Var f_m:fil;f_t1:txt);

Begin

write('введите имя файла ');

writeln('для записи исходного массива');

writeln(' в файл типа mas');

write('nam_m '); readln(nam_m);

write('введите имя файла ');

writeln('для записи исходного массива');

writeln(' в файл типа text');

write('nam_t '); readln(nam_t);

assign(f_m,nam_m); rewrite(f_m);

assign(f_t1, nam_t); rewrite(f_t1);

write('ввод значений элементов');

writeln(' одномерного массива');

writeln(f_t1,'исходный массив');

for i:=1 to n do

begin

write('введите ',i, ' элемент массива'); readln(m1[i]);

write(f_m,m1); {запись элементов массива в файл mas}

```

        write(f_t1,m1[i]:7:1); {запись элементов массива в файл типа text}
    end;
    writeln(f_t1);
    close(f_m);close(f_t1);
end;

```

```

{процедура чтения массива из файла типа mas и вывода его на экран}
PROCEDURE Read_Mas(nam_m:filename;var m1:mas);
Begin
    write('введите имя файла для чтения nam_m '); readln(nam_m);
    assign(f_m,nam_m);
    reset(f_m); {процедура открывает файл типа mas для чтения}
    for i:=1 to n do
        begin
            read(f_m,m1); {чтение элементов массива из файла типа mas}
            write(m1[i]:3:1, ' '); {запись элементов массива на экран}
        end;
    close(f_m);writeln;
end;

```

```

{процедура формирования нового массива из положительных значений эл-ов}
PROCEDURE Form_Mas(var m1,m2:mas; var k:integer);
Begin
    k:=0;
    for i:=1 to n do
        begin
            if m1[i]>=0 then
                begin k:=k+1; m2[k]:=m1[i]; end;
            end;
        end;
end;

```

```

{процедура дозаписи нового массива в новый файл типа text}
PROCEDURE Zapis_Newmas(nam_t:filename;var m2: mas;
    var f_t1:txt; Var k:integer);
Begin
    writeln('введите имя файла для записи нового массива');
    writeln('в файл типа text – nam_t '); readln(nam_t);
    writeln(f_t1);

```

```

append(f_t1); {процедура открывает файл для дозаписи нового массива}
writeln(f_t1, 'новый массив');
for i:=1 to k do
    write(f_t1, m2[i]:7:1);
writeln(f_t1); close(f_t1);
end;

```

{главная программа}

```

BEGIN
    clrscr;
    zapis_mas(nam_m, nam_t, m1, f_m, f_t1);
    read_mas(nam_m, m1); form_mas(m1, m2, k);
    zapis_newmas(nam_t, m2, f_t1, k);
    ch:=readkey
END.

```

Лабораторная работа № 11

ДВУХМЕРНЫЕ МАССИВЫ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования двухмерных массивов.

Задачи: построение алгоритмов с использованием и обработкой двухмерных массивов.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Массивы массивов. Матрицы

Массивы бывают одномерными и многомерными. Количество индексов в описании массива указывает на его размерность. Для того, чтобы обратиться к элементу такого массива, нам необходимо, после имени массива указать все индексы этого элемента в квадратных скобках через запятую. Например, массив с именем C4 имеет размерность, равную 4.

Type

```

C1=array [1..5] of integer;
C2=array [1..4] of C1;
C3=array [1..5] of C2;
C4=array [1..10] of C3;

```

Var mas: c4;

или

C4: array [1..5,1..4,1..5,1..10] of integer;

Для того, чтобы обратиться к элементу массива mas, необходимо, после имени массива указать четыре индекса элемента, например: mas[2,3,5,1].

Особый интерес представляют *двумерные массивы*, которые ещё называют квадратными и прямоугольными таблицами. В научной литературе их часто называют матрицами, при этом элементы матриц изображаются с помощью двух индексов.

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

или

$$\begin{vmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{vmatrix}$$

Первый индекс – номер строки, второй – номер столбца; a_{13} читается "а один три", а не "а тринадцать". В программах на Паскале используются соответственно конструкции $a[1,3]$.

Про матрицу, имеющую m строк и n столбцов, говорят, что она имеет размер $m \times n$ ("m на n"). Если $m=n$, то матрица называется квадратной.

Для квадратной матрицы элементы с одинаковыми индексами для строк и столбцов составляют главную диагональ матрицы – это элементы a_{11} , a_{22} , a_{33} . Противоположная диагональ называется побочной диагональю. К ней относятся элементы a_{13} , a_{22} , a_{31} .

Существуют две формы описания двухмерного массива, с помощью Var или Type и Var.

Первая форма описания двухмерного массива – описание матрицы размером 20 строк на 20 столбцов, содержащего 400 элементов типа byte:

Var Matr:array[1..20,1..20] of byte;

Вторая форма описания – описание матриц с именами Matr1 и Matr2, размером 25 строк и 10 столбцов, содержащего 250 элементов типа real:

```
Type Tmatr = Array[1..25,1..10] of real;  
Matr1, Matr2:Tmatr;
```

Кроме того, двумерный массив можно интерпретировать как одномерный, элементами которого является другой одномерный массив. Описание такого массива равносильно описанию, приведенному выше для массивов с именами Matr1 и Matr2.

```
Type tstr=array[1..25] of real;  
Var masssiv:array[1..10] of tstr;
```

Если массив двумерный, то память под него выделяется так, что быстрее меняется самый правый индекс. Если массив объявлен, то к любому его элементу можно обратиться с помощью его имени и индексов. В качестве примера рассмотрим порядок выделения оперативной памяти под массив, описанный следующим образом:

```
Var M:array[1..2,1..4] of byte;
```

Этот массив будет располагаться в памяти в следующем порядке: M[1,1]; M[1,2]; M[1,3]; M[1,4]; M[2,1]; M[2,2]; M[2,3]; M[2,4].

Обработка матриц

Для обработки матриц наиболее удобно применять вложенные циклы с параметром.

Заполнение матрицы на 400 чисел случайными числами из диапазона от 0 до 100:

```
For i:=1 to 20 do  
  For j:=1 to 20 do  
    Matr[i, j]:=random(100);
```

Заполнение матрицы целых чисел размером 4x5 с клавиатуры:

```
For i:=1 to 4 do  
  For j:=1 to 5 do  
    begin  
      write('Введите элемент ',i,' ',j,' ');  
      readln a[i, j];  
    end;
```


Операция присваивания.

В системе программирования TP7.0, для одинаковых массивов допустима операция присваивания массива массиву. Например, если описано два массива следующим образом:

```
VAR A, B: array[1..50,1..60] of real;
```

то допустима следующая операция присваивания: A:=B;

Задача 26: дана матрица действительных чисел размером 3x5. Вычислить сумму элементов матрицы.

```
PROGRAM Lab11_1;
  CONST
    n=3; {Количество строк}
    m=5; {Количество столбцов}
  TYPE
    mas=array[1..n,1..m] of real;
  VAR
    b: mas; {Матрица размером n x m}
    i: integer; {Индекс строки}
    j: integer; {Индекс столбца} s: real; {Сумма}
    {Процедура ввода значений матрицы с клавиатуры}
  PROCEDURE Vvod_Mas(Var b:mas);
  Begin
    Writeln('Введите значения элементов матрицы: ');
    for i:=1 to n do
      for j:=1 to m do
        begin
          write('Введите элемент ',i,' ',j,' '); readln(b[i,j]);
        end;
      end;
    end;
    {Процедура вывода значений элементов матрицы на экран}
  PROCEDURE Vyvod_Mas(Var b:mas);
  Begin
    for i:=1 to n do
      begin
        for j:=1 to m do write(b[i, j], ' '); writeln
        end;
      end;
    End;
```

{Процедура вычисления суммы}

```
PROCEDURE Sum_Mas(Var b:mas; Var s:real);
```

```
Begin
```

```
    s:=0;
```

```
    for i:=1 to n do
```

```
        for j:=1 to m do
```

```
            s:=s+b[i, j];
```

```
End;
```

```
BEGIN
```

```
    Clrscr;
```

```
    Vvod_Mas(b);
```

```
    Vyvod_Mas(b);
```

```
    Sum_Mas(b,s);
```

```
    writeln('Сумма = ',s);
```

```
    Readln
```

```
END.
```

Задача 28: дана матрица целых чисел размером 4x4. Заменить все элементы главной диагонали нулями. Вывести на экран монитора содержимое матрицы до обработки и после обработки.

```
PROGRAM LAB11_2;
```

```
CONST n=4;
```

```
Type
```

```
    mas= array[1..n,1..n] of integer {Матрица}
```

```
VAR
```

```
    a:mas;
```

```
    i, j: integer;
```

```
PROCEDURE Vvod_Mas(Var a:mas);
```

```
Begin
```

```
    for i:=1 to n do
```

```
        for j:=1 to n do
```

```
            begin
```

```
                write('Введите элемент ',i,' ',j,' '); readln a[i, j];
```

```
            end;
```

```
End;
```

```

PROCEDURE Vyvod_Mas(Var a:mas);
Begin
  for i:=1 to n do
    begin
      for j:=1 to n do write(a[i, j], ' '); writeln
      end;
    end;
End;

```

```

PROCEDURE Form_New_Mas(Var a:mas);
Begin
  for i:=1 to n do
    a[i ,i]:=0; {Обнуление диагональных элементов матрицы}
  End;

```

```

BEGIN
  Clrscr;
  Vvod_Mas(a);
  Writeln('Содержимое матрицы до обработки ');
  Vyvod_Mas(a);
  Form_New_Mas( a);
  writeln('Содержимое обработанной матрицы ');
  Vyvod_Mas(a);
  Readln
END.

```

Задача 29: составить программу определения сумм элементов в каждой строке матрицы $M=\{m_{ij}\}$, $i=1, n$; $j=1, k$, где n – число строк, k – число столбцов матрицы, m_{ij} – целые числа из диапазона: от 0 до 500. Записать полученные значения сумм в одномерный массив SUM , а затем вывести их на экран дисплея. Числа в массив M занести с помощью функции **Random**. Определить и вывести на экран номер строки с минимальным значением суммы.

```

PROGRAM Lab11_3;
  Uses crt;
  Var
    M:array[1..50,1..100]of integer;
    Sum:array[1..50] of longint;
    n, k, i, j, шmin:integer; Min:longint;

```

```

BEGIN
  ClrScr;
  writeln(' Введите число строк и столбцов'); readln(n,k);
  Randomize;
  {Заполнение матрицы случайными числами}
  for i:=1 to n do
    for j:=1 to k do
      M[i,j]:=Random(500);
    writeln(' Элементы заполненной матрицы');
  for i:=1 to n do
    begin
      for j:=1 to k do
        write(M[i,j]:4);
      writeln;
    end;
  writeln(' Сумма элементов в каждой строке');
  write(' Номера строк : ');
  for i:=1 to n do write(i, ' ');
  writeln; write(' Сумма в строке : ');
  for i:=1 to n do
    begin
      Sum[i]:=0;
      for j:=1 to k do
        Sum[i]:=Sum[i]+M[i,j];
      write(Sum[i], ' ');
    end;
  writeln;
  {Поиск минимального значения}
  Min:=Sum[1]; imin:=1;
  for i:=1 to n do
    if Sum[i] <= Min then begin Min:=Sum[i]; imin:=i; end;
  writeln(' Минимальная сумма = ',Min,' в строке ',imin);
  Readln;
END.

```

Запись двумерных массивов в файл и чтение из файла

<i>Запись введенной матрицы в файл типа массив</i>	<i>Запись введенной матрицы в текстовый файл</i>
<pre> Const m=2;n=3; Type matr=array[1..m,1..n] of re- al; f=file of matr; var m1:matr; fm:f; l, j: integer; Begin assign(fm,name_file_massiv'); rewrite(fm); for i:=1 to m do for j:=1 to n do\ write(fm,m1); close(fm); end.</pre>	<pre> Const m=2;n=3; Type matr=array[1..m,1..n] of re- al; txt=text; var m1:matr; ft:txt; l, j:integer; Begin assign(ft, 'name_file_txt'); rewrite(ft); for i:=1 to m do begin for j:=1 to n do write(ft,m1[l,j]:5:1); writeln(ft) end; close(ft); end.</pre>
<i>Чтение элементов матрицы из файла типа массив</i>	<i>Чтение элементов матрицы из текстового файла</i>
<pre> Const m=2;n=3; Type matr=array[1..m,1..n] of real; f=file of matr; var m1:matr; fm :f; l,j:integer; Begin assign(fm, 'name_file_massiv'); reset(fm); for i:=1 to m do for j:=1 to n do read(fm,m1); close(fm); end.</pre>	<pre> Const m=2;n=3; Type matr=array[1..m,1..n] of real; txt=text; var m1:matr;t :txt; l,j: nteger; Begin assign(ft, 'name_file_txt'); reset(ft); for i:=1 to m do begin for j:=1 to n do read(ft,m1[l,j]); readln(ft); end; close(ft) end.</pre>

Дозапись элементов матрицы ≤ 0 в текстовый файл

```
Const m=2;n=3;
Type
  matr=array[1..m,1..n] of real;
  txt=text;
var
  m1:matr;
  ft:txt; l, j: integer;
Begin
  assign(ft, 'name_file_txt');
  Append(ft);
  for i:=1 to m do
    for j:=1 to n do
      if m2[l,j]<=0 then
write(ft,m1[l,j]:5:1);
    close(ft);
  end.
```

Лабораторная работа № 12

СИМВОЛЬНЫЕ, СТРОКОВЫЕ ПЕРЕМЕННЫЕ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования символьных и строковых переменных.

Задачи: построение схем алгоритмов с использованием символьных и строковых переменных и запись их в файл.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Символьные переменные. Тип данных "CHAR"

Переменные, предназначенные для хранения одиночных символов, называются символьными переменными. В языке TP для них определён тип данных *CHAR*. В переменную этого типа может быть помещён любой из 256 символов расширенного кода ASCII.

Таблица расширенного кода ASCII

Объявление символьных переменных осуществляется в разделе объявления переменных с помощью служебного слова *CHAR*. Этот тип данных обладает некоторыми особенностями. Обычно значения для переменных типа "*CHAR*" задаются в апострофах. Например, если в программе есть описания *u, v: char*, то возможны операторы присваивания *u:='a'; v:=u; v:='** и т.д.

Штрих ' – принятая в Паскале форма кавычки - употребляется всякий раз, когда значение типа "*CHAR*" явно указывается в программе. Выполнение операторов *u:='b'; write(u)* приводит к высвечиванию на экране символа *b*.

Кроме того, имеется возможность задавать значения указанием непосредственно числового значения ASCII-кода. В этом случае Вы должны перед числом, обозначающим код символа ASCII, поставить знак "#" (*u:=#97*)

Задача 30: дана последовательность символов, заканчивающаяся символом "!". Необходимо подсчитать количество восклицательных знаков в данной последовательности.

```
PROGRAM Lab13_1;
```

```
  Var
```

```
    c: char;
```

```
    n: integer;      {переменная для хранения количества "!"}
```

```
BEGIN
```

```
  read(c);          {чтение первого символа с клавиатуры}
```

```
  n:=0;             {обнуление счетчика "!"}
```

```
  while c<>'/' do {условие окончания работы ввода символа '/'}
```

```
    begin
```

```
      if c='!' then n:=n+1;      {подсчёт символов "!"}
```

```
      read(c);                   {чтение следующего символа}
```

```
    end;
```

```
END.
```

Обратите внимание. В таблице ASCII большие латинские буквы и малые латинские буквы расположены подряд по алфавиту. Такая упорядоченность открывает возможность использования в программах операторов цикла с параметром, имеющим тип "*CHAR*". Параметр цикла пробегает последовательность символов в указан-

ных границах. Выполнение оператора цикла *for c:='a' to 'z' do write(c)*, где *c* символьная переменная, приводит к высвечиванию на экране последовательности всех малых букв латинского алфавита: *abcdefghijklmnopqrstuvwxy*

Выполнение оператора цикла *for c:='z' downto 'a' do write(c)* приведёт к выводу этих же букв в обратном порядке. Символьные переменные очень удобны для написания диалоговых программ. Представим себе, что вам необходимо довольно часто получать значения некоторой функции, например значение синуса от корня квадратного из *x* для разных значений *x*.

Задача 31: программа, в ходе выполнения которой на экране регулярно возникает вопрос, следует ли продолжать работу, или прекратить выполнение программы:

```
PROGRAM Lab13_2;
  Var x: real; s: char;
BEGIN
  s:='д';
  while s='д' do
    begin
      write('x=);read(c);
      writeln(' ,f(x)='sin(sqrt(x)));
      write('продолжить?(д или н )');
      readln(s); write(' ');
    end;
END.
```

Когда выполняется эта программа, на экране возникает следующая последовательность строк:

```
x=...,f(x)=...
продолжить?(д или н) д
. . . .
x=...,f(x)=...
продолжить?(д или н) н
```

С вводом буквы "н" выполнение программы заканчивается. Возможность прочтения буквы ответов обеспечивается использованием в программе символьной переменной.

Функции для работы с символьными переменными

CHR(x: byte): char; Преобразует целое число, имеющее тип BYTE, в один символ ASCII-кода. Следующая программа иллюстрирует возможности функции "CHR". Она выдаёт все символы кода ASCII на экран. При этом некоторые символы ASCII-кода (например, 7, 10, 13) при обычных условиях не имеют изображения, а используются для реализации специальных функций (перевод курсора и т.д.) Благодаря использованному в операторе *Write* формату каждый из изображаемых символов отделяется от соседнего символа пробелом (занимает две позиции).

```
PROGRAM Lab13_3;  
  Var i: byte;  
BEGIN  
  for i:=0 to 255 do write(chr(i):2);  
END.
```

ORD(c: char): byte; Функция *Ord* выполняет действие, обратное функции *Chr*, т.е. возвращает порядковый номер символа параметра в таблице ASCII.

UpCase(c: char): char; Осуществляет преобразование символов английского алфавита из строчных символов в прописные символы. Все остальные символы при применении этой функции остаются непреобразованными.

Символьные массивы. Строки

На сегодняшний день значительные мощности вычислительной техники ми огромное количество разнообразного программного обеспечения используются для работы с текстовой информацией. По этой причине в программировании создаются специальные средства для работы с текстом, разрабатываются соответствующие приёмы и методы программирования. Данная лабораторная работа посвящена изучению средств системы программирования TP, предназначенных для обработки фрагментов текста - строк.

Наиболее простым способом для работы с текстом является применение символьных переменных и линейных массивов, содержащих данные символьного типа (char): VAR ms: array[1..80] of CHAR;

В этом случае необходимо использовать все правила, приёмы и методы обработки массивов. Но как показывает практика, такой подход не совсем удобен. Поэтому для работы с текстом в языке TP был введён специальный тип данных, предназначенный для работы с фрагментами текста – *строками* (*цепочками символов*).

Вам уже неоднократно приходилось в различных ситуациях пользоваться строками для вывода информации с помощью операторов `Write` и `Writeln`: `Writeln('Текст заключённый в кавычки - это и есть строка.');`

Символ номер 32 "кавычка" используется при работе со строковыми величинами так же, как и с символьными переменными. Если Вам необходимо вывести символ номер 32 на экран, перед ним необходимо поставить дополнительную кавычку: `Writeln('Строка с кавычкой " внутри.');` При выполнении этого оператора на экране монитора вы получите: *Строка с кавычкой ' внутри.*

Операции над строками

Объявление строчных типов и переменных. Для объявления строковых типов и переменных используется служебное слово *STRING*, вслед за которым в квадратных скобках указывается максимальная длина строки.

```
VAR s1: string[70]
или
TYPE Line = string[70];
VAR s1: Line;
```

В приведённых выше примерах переменная *s1* в качестве своего значения может иметь любую последовательность символов (каждый из которых имеет тип *char*) произвольной длины (от 0 до 70 символов).

Определения значения строковой переменной. Значение строковой переменной может быть присвоено оператором присваивания, либо введено оператором ввода: `s1 := 'Пример строки.;`
`Readln(s1);`

В случае присваивания строковой переменной строкового выражения с длиной большей, чем максимально допустимая длина для данной переменной, происходит "обрубание" строки до максимальной длины. Эта ситуация не считается ошибочной, поэтому прерывания выполнения в данном случае не происходит.

```

PROGRAM Lab13_4;
VAR
  ShortStr: string[5];
BEGIN
  ShortStr := 'Очень длинная строка';
  Writeln (ShortStr);
END.

```

Операция конкатенации. Для строковых величин определена операция конкатенации '+':

```

PROGRAM Lab13_5;
VAR s1: string[80];
BEGIN
  s1 := 'Пример ' + 'строки';
  Writeln(s1);
END.

```

Длина строки. В TP максимальная длина строки не может превышать 255 символов. Если размер строки не указан, он считается равным 255. Ниже объявлены две строки одинаковой длины. *VAR att: string; ts2: string[255];*

Важнейшее отличие строк от обычных символьных массивов заключается в том, что строки могут динамически изменять свою длину: если после присваивания *att := 'Короткая строка';* длина строки составит 15 символов, то следующее присваивание *att := att + 'стала длиннее';* увеличит её длину до 29 символов.

Механизм хранения строк. Механизм динамических строк реализован в TP весьма просто, поэтому память под переменные строкового типа отводится по максимуму, а используется лишь часть этой памяти, реально занятая символами строки в данный момент. Более точно для описания строковой переменной длиной N символов отводится N+1 байтов памяти, из которых N байтов предназначены для хранения символов строки, а один байт - для значения текущей длины этой строки. Это можно объяснить с помощью следующего рисунка.

Элементы строки нумеруются, начиная с единицы. Элемент строки с номером 0 используется для хранения текущей длины

строки. Этот элемент тоже доступен для изменения, т.е. возможны прямые присвоения: $S[0] := Chr(7)$; или $S[0] := \#7$;

В таких присвоениях в качестве присваиваемой величины может выступать целое число в диапазоне от 0 до 255, преобразуемое к типу CHAR с помощью префикса "#" или функции "Chr". Такое преобразование необходимо, так как каждый элемент строки, в том числе и показатель длины должен быть символом "Char". Обращение к нулевому элементу строки можно использовать для "принудительного" изменения длины строки, например:

```
PROGRAM Lab13_6;
  VAR W: string[60]; l: byte;
BEGIN
  W := '-----';
  for l:=1 to 12 do
    begin
      W[0] := Chr(l*5); Writeln (W);
    end;
END.
```

Отдельные элементы строк. Доступ к отдельным элементам строк производится аналогично доступу к элементам одномерного массива: после имени строковой переменной необходимо в квадратных скобках указать номер элемента строки. Данная конструкция имеет символьный тип CHAR и является обычной переменной, которая может стоять слева от знака операции присваивания. Однако, определённую аналогию строковых типов с символьными массивами нельзя понимать как их полную идентичность. Так, распространённой ошибкой является работа с элементами строки без учёта её текущей длины, что иллюстрирует следующий пример программы:

```
PROGRAM Lab13_7;
  VAR
    Str: string[26]; i: integer;
BEGIN
  Str := 'A';
  for i:=1 to 26 do Str[i]:=Chr(Ord('A')+i-1);
  Writeln(Str);
END.
```

Предполагается, что данная программа должна сформировать строку из 26 символов, содержанием которой является последовательность заглавных букв латинского алфавита. Однако вызов процедуры *Writeln* покажет, что содержанием переменной *Str* будет строка из одного символа *A*. Ошибка заключается в том, что присваивание значений элементам строки *НЕ ВЛИЯЕТ* на её текущую длину, которая была установлена равной 1 при первом присваивании (здесь значение получала строка в целом). Поэтому более правильной будет следующая программа:

```
PROGRAM Lab13_8;  
  VAR  
    Str: string[26]; i: integer;  
BEGIN  
  Str := "";  
  for i:=1 to 26 do  
    Str := Str+Chr(Ord('A')+i-1);  
  Writeln(Str);  
END.
```

Функции для работы со строками

Concat (S1, [S2, ..., Sn]: string): string; Объединяет несколько строк в одну (при необходимости усекает чрезмерно большую строку до 255 символов). *S1, S2, ..., Sn* - объединяемые строки.

Length (S: string): integer; Возвращает текущий размер строки. *S* – строка, у которой определяется размер.

```
PROGRAM Lab13_9;  
  VAR W:string;  
BEGIN  
  Write('Введите, пожалуйста, слово'); Readln(W);  
  Writeln('Это слово состоит из ',Length(W),'букв! ');  
END.
```

Pos (P,S: string): byte; Поиск последовательности *P* в строке *S* (результат равен номеру первого символа строки *S*, с которого начинается искомая последовательность, или 0, если такой последовательности в строке нет).

Процедуры для работы со строками

Copy (*S*: string; *I*: integer; *C*: integer); Создает подстроку строки **S**, где **S** - исходная строка; **I** - номер первого, выделяемого символа строки, (если значение больше длины строки, возвращается пустая строка); **C** - число выделяемых символов (если всех необходимых символов в строке нет, возвращается имеющийся остаток строки).

```
PROGRAM Lab13_10;
```

```
VAR
```

```
W: string[79];  
s1, s2, s3: string[20];
```

```
BEGIN
```

```
W:= 'картографирование'; Writeln(W); {картографирование}  
s1 := Copy(W,6,4);           Writeln(s1);           {граф}  
s2 := Copy(W,2,3);           Writeln(s2);           {арт}  
s3 := Copy(W,11,3);          Writeln(s3);           {ров}
```

```
END.
```

Delete (*var S*: string; *I*: integer; *C*: integer); Удаляет подстроку из строки **S**, где **I** - номер первого удаляемого символа (если номер больше размера строки, символы не удаляются); **C** - число удаляемых символов (если символов в строке недостаточно, удаляется остаток символов).

Insert (*P*: string; *var S*: string; *I*: integer); Вставляет подстроку **P** в строку **S** (если получается строка слишком большого размера, то она усекается до 255 символов), где **S** - исходная строка; **P** - подстрока, помещаемая в строку; **I** - номер позиции исходной строки, начиная с которой помещается подстрока.

```
PROGRAM Lab12_11;
```

```
VAR s1: string[79]; s2: string[20];
```

```
BEGIN
```

```
s1 := 'компьютеризация'  
Writeln(s1);                               {компьютеризация}  
Delete(s1,1,7);                             Writeln(s1);           {еризация}  
Delete(s1,3,2);                             Writeln(s1);           {ерация}  
s2 := 'Г'; Insert(s2,s1,1); Writeln(s1);     {Герация}  
s2 := 'не'; Insert(s2,s1,3); Writeln(s1);     {Генерация}
```

```
END.
```

Задача 31: ввести строку символов с клавиатуры и записать в файл. Прочитать из файла строку. Заменить символом пробел символы " ! , . ? ", содержащиеся в строке. Дописать в файл новую строку после замены.

```
PROGRAM Lab12_12;
  Uses crt;
  TYPE
    ff=text; s=string;
  Var
    str,s1:s;
    l:integer;
    f:ff;
    s2:char;

  PROCEDURE Zapf(var str:s);
  Begin
    rewrite(f);
    writeln('Введите строку'); readln(str);
    writeln(f,str);
    close(f);
  end;

  PROCEDURE Readf(var str:s);
  Begin
    reset(f);
    readln(f,str);
    close(f);
  end;

  PROCEDURE Zamena(var str,s1:s);
  Begin
    s1:=' ';
    for l:=1 to length(str) do
      if (str[l]=',') or (str[l]='!') or (str[l]='.') or (str[l]='?') then
        begin
          delete(str,l,1); insert(s1,str,l);
        end;
    writeln(str);
  end;
```

```

PROCEDURE Dozap(var str:s);
Begin
    Append(f);
    writeln(f,str);
    close(f);
end;

```

```

BEGIN
    clrscr;
    Assign(f,'myfile');
    Zapf(str);
    Readf(str);
    Zamena(str,s1);
    Dozap(str);
    readln

```

END.

Задача 32: вставляет '!!!' после буквы 'E'}

```

PROGRAM Lab12_13;
Uses Crt;
var
    i:integer; ch:char;
    k:string[1]; str:string[25];
    ss,s1:string;
BEGIN
    clrscr;
    writeln('ввести строку');
    readln(str);
    k:='e';ss:='';s1:='!!!';
    for i:=1 to length(str) do
        begin
            if str[i]<>k then ss:=ss+str[i]
                else ss:=ss+str[i]+s1;
            end;
        writeln('str=',str);
        writeln('nov=',ss);
        ch:=readkey
    END.

```


Лабораторная работа № 13

ЗАПИСИ

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования записей: научиться описывать записи с заданной структурой; освоить приемы ввода данных по полям записи

Задачи: построение схем алгоритмов с использованием записей, формирование файла из записей.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тип данных – записи, их описание и использование

Запись - это структура данных, состоящая из фиксированного числа компонентов, называемых полями. Запись имеет имя. Каждое поле записи также имеет имя. Обращение к любому элементу записи осуществляется по составному имени, имеющему вид:

Имя записи. Имя поля

В отличие от массива компоненты (поля или элементы) записи могут быть различного типа. Записи удобно использовать в тех случаях, когда необходимо описать атрибуты (характеристики или свойства) одного объекта, имеющие различный тип.

Запись может быть объявлена в разделе *Var* или в разделах *Type* и *Var*, одновременно.

Var

```
Имя записи:RECORD
    Имя поля 1:тип;
    Имя поля 2:тип;
    ...
    Имя поля n:тип
END;
```

Type

```
Имя типа для записи
=RECORD
    Имя поля 1:тип;
    Имя поля 2:тип;
    ...
    Имя поля n:тип
END;
Var Имя записи: Имя типа для
записи;
```

Второй способ описания более предпочтителен, будем использовать его.

Пример необходимо описать список студентов, имеющий следующую структуру:

№ п/п	ФИО	Факультет	Группа	Экзаменационные оценки Физика Математика Програм- миров.
-------	-----	-----------	--------	--

Описание представленной структуры будет иметь следующий вид:
Type

```
T_Stud=Record
  Nom:word;
  FIO:string[40];
  Fac:string[5];
  Group:string[6];
  Ball:array[1..3] of byte;
End;
```

```
Var Stud:T_Stud; {Описание одной записи}
```

Записи могут быть составными частями массивов или других записей. Так для описания всего списка студентов, состоящего, допустим, из 1000 записей необходимо записать:

```
Spisok:Array[1..1000] of T_Stud; {Описание массива из записей}
```

Элементы записи (поля) используются в программе как обычные переменные данного типа. Чтобы обратиться к i -й записи представленного списка (массива *Spisok*) и к его полю *FIO* необходимо записать: *Spisok[i].FIO*. Для того, чтобы обратиться к той же записи и выбрать оценку по математике необходимо записать *Spisok[i].Ball[2]*.

Оператор присоединения.

Запись в файл типа данных – запись

Для упрощения обращения к элементам записи используется оператор присоединения

```
WITH Имя записи DO
```

```
  Begin
```

```
    <Операторы обращения к элементам записи>
```

```
  End;
```

Для нашего примера используем этот оператор при суммировании трех оценок i -го студента:

```

. . . .
Sum:=0;
With Spisok[i] do
  Begin
    For j:=1 to 3 do
      Sum:=Sum+Ball[j];
    . . . .
  End;

```

Рассмотрим пример использования записи внутри другой записи. Имеется структура:

№ п/п	Ф.И.О.	Дата рождения			Пол	Национальность
		число	месяц	год		

Поле дата рождения состоит из полей: число, месяц, год. Описание в программе будет иметь вид:

```

Type
  Date=Record
    Days:1..31;
    Month:1..12;
    Year:1900..2000;
  End;

Tzap=Record
  Nom:word;
  FIO:string[40];
  Birthday:Date;
  Pol:(Man,Woman);
  Nac:string[20];
  End;

Var Zap:Tzap;

```

Для того чтобы обратиться к элементам записи дата рождения, необходимо записать:

```
D:=Zap. Birthday.Days;  
M:= Zap. Birthday.Month;  
G:= Zap. Birthday.Year;
```

или с использованием оператора присоединения

```
With Zap. Birthday do  
Begin  
  D:=Days; M:=month; G:=Year;  
End;
```

Типизированные константы - записи имеют вид:

Имя константы : Тип = (Список значений полей записи);

Список значений полей записи - это список из последовательностей вида: Имя поля: Константа;

Пример:

```
Type  
Tdate=Record  
  Day:1..31;  
  Month:1..12;  
  Year:1000..2000;  
End;  
Const  
  Date:Tdate=(day:25;month:10;year:1999);
```

Пример:

```
Type  
Tkoord=record  
  Xk, Ykreal;  
end;  
Tmas=array[1..3] of Tkoord;  
Const  
  Maskoord:Tmas=((Xk:0.0; Yk:0.0),  
                  Xk:1.5; Yk:2.5),  
                  Xk:3.0; Yk:4.5));
```

Работа с массивом из записей

Задача 33: дан список, содержащий N записей ($N \leq 100$) следующей структуры:

№ рейса	Пункт отправления	Пункт назначения	День недели	Время отправления		Цена билета
				Час	Мин	
5 символов	15 символов	15 символов	1..7	0..23	0..59	Real

1. Ввести заданный список с экрана в массив записей *Spis*.
2. Ввести искомый номер рейса - *Isk_nom*.
3. Найти в списке рейс с заданным номером.
4. Вывести информацию о найденном рейсе на экран.

PROGRAM Lab13_1;

Uses Crt;

Type

T_Time=record

Hour:0..23; Min:0..59;

end;

tzap=record {Описание типа для одной записи списка}

nom:string[5]; p1, p2:string[15];

day:1..7; time:T_Time; price:real;

end;

Var

Spis: array[1..100] of tzap; {Описание списка}

N, i:byte; Isk_nom:string[5];

BEGIN

Write(' Введите число записей в списке N = '); Readln(N);

Write('Введите список по полям – ');

Writeln(' каждое поле с новой строки');

For i:=1 to N do

With Spis[i] do

begin

Write('Номер очередного рейса - 5 символов ');

Readln(nom);

```

Write('Пункт отправления - 15 символов ');Readln(p1);
Write(' Пункт назначения - 15 символов ');Readln(p2);
Write(' День недели - от 1 до 7 ');Readln(day);
Write('Время отправления - часы (от 0 до 23) и');
Write(' минуты (от 0 до 59), через пробел ');
Readln(time.hour,time.min);
Write(' Цена билета = ');Readln(price);
end;
write('введите искомый номер ');
writeln(' рейса - 5 символов'); Readln(Isk_nom);
{Организация поиска заданного рейса в массиве записей Spis}
for i:=1 to N do
  with Spis[i] do
    if nom = Isk_nom then
      begin
        writeln(' Рейс найден'); write(nom,' ',p1,' ',p2,' ');
        writeln(day,' ',time.hour,',',time.min,',',price); readkey;
        halt;
      end;
    writeln(' Рейс не найден'); readkey;
  end;
END.

```

Примечание: При вводе символьной информации необходимо учитывать, что пробел также является символом.

```

PROGRAM Lab13_2;
uses crt;
const
  n=3;
type
  Books=RECORD
    Nomer:integer; Avtor:STRING[15];
    Nazv:string[10];
  END;
var
  BooksFile:text;
  Rbooks:books;
  Filename:string[25];

```

```

ch:char; i,rej:integer;
Procedure ORG;
Begin
  write('MMH файла ');
  readln(fileName);
  assign(Booksfile, Filename);
  rewrite(Booksfile);
  with Rbooks do
    while True do
      begin
        write('Введите номер книги ');
        readln(Nomer);
        if Nomer=9999 then
          begin
            Close(BooksFile); Exit
          end;
        write('Введите автора книги '); readln(Avtor);
        write('Введите название книги '); readln(Nazv);
        writeln(BooksFile,nomer:10,avtor:15,nazv:10)
      end;
    End;
Procedure OBR;
Begin
  writeln(' ввести имя файла:');
  readln(Filename);
  assign(BooksFile, Filename);
  reset(BooksFile);
  with Rbooks do
    while not Eof(BooksFile) do
      begin
        read(BooksFile,nomer,Avtor,Nazv);
        writeln(Nomer:10,Avtor:15,Nazv=10)
      end;
    close(booksfile);
  End;

```

```

Procedure Rash;
Begin
  write(' ввести имя исх. файла ');
  readln(Filename);
  assign(BooksFile,Filename)
  reset(BooksFile);
  append(BooksFile);
  with Rbooks do while true do
    begin
      write('ввести значение Nomer '); readln(Nomer);
      if Nomer = 9999 then
        begin
          close(BooksFile);
          Exit
        end;
      write('ввести значение Avtor '); readln(Avtor);
      writel'ввести значение Nazv '); readln(Nazv);
      writeln(BooksFile,Nomer:10, Avtor:15,Nazv:10)
    end;
  End;
BEGIN
  Clrscr,
  Org;
  Obr;
  Rash;
  ch:=readkey
END.

```

Лабораторная работа № 14

МНОЖЕСТВА

Цель работы: приобрести начальные навыки работы в системе Turbo Pascal на примере программирования множеств.

Задачи: построение схем алгоритмов с использованием множеств, запись их в файл.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тип "множество"

Этот тип является одним из сложных типов данных системы программирования TP7.0.

Под *множеством* в языке Паскаль понимают ограниченный неупорядоченный набор различных элементов одинакового типа, логически связанных друг с другом. Количество элементов, входящих в множество, может меняться в пределах от 0 до 255. Множество, не содержащее элементов, называется пустым. Множество имеет имя. Тип элементов, входящих во множество, называется базовым. В качестве базового типа можно использовать любой порядковый тип, кроме *Word*, *Integer*, *Longint*.

Описание (декларация) типа множество

Множества должны быть объявлены либо в разделе *Var*, либо в разделах *Type* и *Var*, одновременно. Для задания типа "множество" следует использовать служебные слова **SET** и **OF**, а затем указать элементы этого множества, как правило, в виде перечисления или диапазона,

Var Имя множества:Set of базовый тип;

или

Type Имя типа=Set of базовый тип;

Var Имя множества:Имя типа;

Type

Alfa = set of 'A'..'Z'; Ten = set of 0..9;

Count = set of (Plus, Minus);

Var

CAlfa: Alfa; U233: Ten;

Так же как и для других структурированных типов, тип множество можно задать непосредственно при задании переменных: *Var CAlfa: set of 'A'..'Z'*; {Множество заглавных латинских букв} *Operation: set of (Plus, Minus)*; *U233: set of 0..9*; {Множество целых чисел от 0 до 9} *C: set of char*;

Присвоение для переменных типа множество

Множеству можно в программе присвоить то или иное значение. Обычно значение задаётся с помощью, так называемого конструктора множества.

Значения переменных множества задаются в разделе операторов с помощью конструктора множества, который представляет собой список элементов базового типа, заключенный в квадратные скобки.

```
CAIpha := ['A', 'B', 'C']; C := [chr(0)..chr(31), 'D', 'F'];
```

В каждое множество включается и т.н. пустое множество [], не содержащее никаких элементов. Конструктор множества можно использовать и непосредственно в операциях над множествами.

```
...
Var M1,M2,M3:set of 1..99;
Begin ...
    M1:=[];           {Множество пустое}
    M2:=[1,3,5,7,9]; {Множество нечетных чисел в первом
    десятке}
    M3:=[2,4,6,8];   {Множество четных чисел в
    первом десятке}
    ...
End.
```

В качестве элементов в изображении множеств допускается использовать константы и выражения, тип которых совместим с базовым типом. Типизированная константа – множество задается в виде правильного конструктора множества, например:

```
Type
Type_month=(Jn, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct,
            Nov, Dec);
TDays=Set of 1..31;
Tmonth=Set of 1..12;
Tsym=Set of 'A'..'Z';
Tmno=Set of Type_month;
Const
SymMno:Tsym=['A','E','I','O','U']; {подмн-во гласных букв}
```

DaysMno:TDays=[1,8,15,22,29]; {подмн-во выходных дней месяца}
Spring_Mes:Tmonth=[3,4,5]; {подмн-во весенних месяцев}
Spring_Month:Tmno=[Mar,Apr,May]; {то же, что и предыдущее}

Операции над множествами

Так же как и в математике в системе программирования TP над множествами применимы следующие операции:

1). + - *объединение множеств*. $C := A+B$; (каждый элемент множества C является элементом либо множества A , либо множества B , т.е. результат содержит элементы первого множества, дополненные недостающими элементами из второго множества); $S4 := S1+S2$; - в $S4$ будет содержаться [1,3,4,6,2,5];

2). - - *разность множеств*. $C := A-B$; (каждый элемент множества C является элементом множества A , но не является элементом множества B , т.е. результат содержит элементы из первого множества, которые не принадлежат второму); $S5 := S1-S2$; - в $S5$ будет содержаться [3,6];

3). * - *пересечение множеств*. $C := A*B$; (каждый элемент множества C является элементом множества A и B одновременно, т.е. содержит элементы, общие для обоих множеств);

Var S1,S2,S3,S4,S5:Set of 1..10; Begin S1:=[1,3,4,6]; S2:=[2,4,5,1];
 $S3 := S1*S2$; - в $S3$ будет содержаться [1,4].

4). = - *проверка эквивалентности* (или равенства) *двух множеств*. $A = B$; (множество A равно множеству B , если каждый элемент множества A является элементом множества B , и наоборот, каждый элемент множества B является элементом множества A . Возвращает *TRUE*, если оба множества эквивалентны, т.е. содержат все одинаковые элементы);

5). <> - *проверка неэквивалентности* (или неравенства) *двух множеств*. $A <> B$; (множество A не равно множеству B , если множество A содержит хотя бы один элемент, не являющийся элементом множества B , или (и) наоборот, множество B содержит хотя бы один элемент, не являющийся элементом множества A . Возвращает *TRUE*, если оба множества неэквивалентны, т.е. содержат неодинаковые элементы а);

6). <= - *проверка вхождения, является ли левое множество подмножеством правого* ($A <= B$; Множество A является подмноже-

ством множества В, если все элементы множества А являются элементами множества В. Возвращает *TRUE*, если первое множество включено во второе (т.е. все элементы первого множества присутствуют также и во втором);

7). *>=* - проверка вхождения, является ли правое множество подмножеством левого (*A >= B*; Действие операции аналогично действию предыдущей операции. Возвращает *TRUE*, если второе множество включено в первое);

8). *In* - проверка принадлежности элемента множеству, т.е. входит ли элемент (слева) в множество (справа). (*C in B*; Эта операция возвращает результат *TRUE*, если элемент (или выражение), стоящий слева принадлежит множеству, указанному справа).

Дополнительно к этим операциям можно использовать две процедуры:

Include - включает новый элемент во множество: *Include(M,elem)*; где *M* - множество элементов некоторого базового типа, а *elem* - элемент того же типа, который необходимо включить в множество *M*.

Exclude - исключает элемент из множества: *Exclude(M,elem)*. В отличие от операций "+" и "-", реализующих аналогичные действия над двумя множествами, эти процедуры оптимизированы для работы одиночными элементами множества и поэтому отличаются высокой скоростью выполнения.

Основным достоинством использования множеств является экономия памяти: внутренне устройство множества таково, что каждому его элементу ставится в соответствие один двоичный разряд (один бит). Если элемент включен в множество, то соответствующий разряд имеет значение 1, в противном случае - 0. Минимальной единицей памяти является 1 байт, содержащий 8 бит, поэтому для хранения множества мощностью 256 элементов выделяется память 32 смежных байта.

Задача 35: из множества целых чисел 1..20 выделить: множество чисел, делящихся на 6 без остатка; множество чисел делящихся на 2 или на 3 без остатка. Вывести содержимое этих множеств на экран.

```
PROGRAM LAB14_1;
```

```
  Var
```

```
    N6 : set of integer;      {множество чисел, делящихся на 6}
```

```

N23 : set of integer; {множество чисел, делящихся на 2 и 3}
k: integer;
BEGIN
  N6 := [];           {обнуление множества N6}
  N23 := [];         {обнуление множества N23}
  For k:=1 to 20 do  {формирование множеств}
    Begin
      if k mod 6 = 0 then N6 := N6 + [k];
      if (k mod 2 = 0) or (k mod 3 = 0) then N23 := N23 + [k];
    end;
  Writeln(' На 6 без остатка делятся числа: ');
  for k:=1 to 20 do
    if k in N6 then write(k:3);
  Writeln;
  Writeln(' На 2 или 3 без остатка делятся числа: ');
  for k:=1 to 20 do
    if k in N6 then write(k:3);
  Readln;
END.

```

Задача 36: ввести строку символов, состоящую из латинских букв цифр и пробелов. Осуществить проверку правильности введенных символов.

```

Program Lab15_2;
  Var
    s: string;
    m: set of char;
    k: integer;
    Done: boolean;
  Begin
    m := ['0'..'9', 'A'..'Z', 'a'..'z', ' '];
    Writeln(' Введите строку: '); Readln(s); done := true;
    for k:=1 to Length(s) do
      if not (s[k] in m) then done := false;
    If done then writeln('Правильная строка. ')
      Else writeln('Неправильная строка. ');
    Readln
  End.

```

ЗАДАНИЯ ПО ПРОГРАММИРОВАНИЮ

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 1

Таблица 1

Внимательно прочитать, а затем выполнить задания!

№	Наименование заданий
1.	<i>Войти в оболочку TP7.</i>
2.	<i>Работа в меню редактора TP7:</i> – войти в главное Меню; – выбрать опцию File; – поперемещать курсор в выпавшем подменю; – вернуться в экранный редактор.
3.	<i>Выполнить задание 2, используя только комбинации клавиш Alt+<выделенные буквы>в меню TP.</i>
4.	<i>Работа с окнами:</i> – открыть новое окно, и затем сразу же его закрыть ¹ ; – открыть два новых окна; – сделать активным окно номер 2, а затем окно номер 1; – закрыть окна; – открыть новое окно; – изменить его размеры и положение на экране; – закрыть открытое окно; – открыть три новых окна, и попытаться их разместить разными способами на экране монитора; – закрыть все открытые окна.
	PROGRAM Lab_1; USES CRT; BEGIN ClrScr; Writeln('Лабораторная работа № 1.');
	Write('Тема:Работа в интегрированной среде');
	Writeln(' Turbo Pascal 7.0.');
	Writeln('Выполнил студент гр.110119 Иванов Е.Л.');
	Write('ПОЗДРАВЛЯЕМ!');
	Writeln(' КОПИРОВАНИЕ ПРОШЛО УСПЕШНО.');

¹ Предупреждаем, окно, которое откроете, сразу становится активным. Оно перекроет окно, в котором размещён текст данной лабораторной работы. Окно № 2 активно, посмотрите на его номер в верхнем правом углу активного окна.

№	<i>Наименование заданий</i>
	Writeln('Для продолжения нажмите клавишу Enter'); Readln; END.
5.	<i>Работа с меню "File":</i> – открыть диалоговое окно "Save File as"; – сохранить файл с именем "NAME".
6.	<i>Работа с меню "Edit":</i> – выделить приведённый ниже текст программы Lab_1; – скопировать выделенный фрагмент программы в "карман"; – открыть новое окно; – начиная с 10-й строки этого окна, вставить текст программы Lab_1; – сохранить с новым именем "Lab1.PAS"; – выполнить задание, используя только комбинации "горячих" клавиш; – удалить текст этого задания любым из известных вам способов; – восстановить этот фрагмент с помощью операции "откатка изменений"; – выполнить действие обратное действию "откатка изменений"; – закрыть все открытые окна.
7.	<i>Работа с меню "Run":</i> – открыть файл NAME.PAS – выполнить программу, посмотреть результат работы программы; – закрыть все открытые окна; – выйти из оболочки TP7.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 2

Вычислить значения функции при заданных параметрах (исходные параметры задать тип: как целые, так и вещественные различным способом: константами, присваиванием каких-либо значений, оператором ввода). Определить необходимые параметры в задании 2.

Таблица 2 – Линейное программирование

№ п/п	Задание 1	Задание 2
1.	$z = -\sqrt{x + \sqrt[3]{y}}$ $z = \frac{1}{\sqrt[3]{\cos^2 \alpha}} + \arcsin \beta$	Даны два целых числа. Найти: их среднее арифметическое и среднее геометрическое.
2.	$z = \frac{-b + \sqrt{\frac{b}{a^3}}}{2a}$ $z = \frac{1}{\sqrt[5]{2 \cos \alpha}} + \operatorname{ctg} \beta^2$	Дано натуральное число n, состоящее из трех цифр. Определить количество сотен и десятков в нем.
3.	$z = \frac{-b + \sqrt{\frac{b^3}{a}}}{a}$ $z = \sqrt{ \arcsin \beta - \cos \alpha^{2/3} }$	Даны два числа. Найти: их сумму, разность, произведение, а также частное от деления первого числа на второе.
4.	$z = \frac{-b^2 + \sqrt{\frac{b}{3}}}{2a^{2/3}}$ $Z = \operatorname{arctg} a^5 - \sqrt{ \cos 3b }$	Найти сумму членов арифметической прогрессии, если известны ее первый член, разность и число членов прогрессии.
5.	$z = \sqrt{\frac{1 + \operatorname{tg} \alpha^2}{\sin b}}$ $z = \frac{-b + \sqrt{\frac{3a}{b^2}}}{a^2}$	Найти произведение средних и крайних цифр четырехзначного числа
6.	$z = 3 \sin(l + m) + c^{1.3}$ $y = \sqrt{ 2 \operatorname{tg} l + \ln(l^3 + m) }$	Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.

Продолжение таблицы 2

№	Задание 1	Задание 2
---	-----------	-----------

п/п		
7.	$x=0,4l^{1,5} + e^{0,2mb}$; $w = \frac{e^{x-y} + \ln(1+x^2)}{tg(y)}$	Даны катеты прямоугольного треугольника. Найти его периметр и площадь.
8.	$n = \sqrt{ e + \cos(x) } \cdot \lg(y)$; $u = \frac{\sqrt{ x+y }}{\log(tg(2))}$	Дан радиус окружности. Найти длину окружности и площадь круга.
9.	$l = \arccos(x+y) + \sqrt{ x \cdot y }$; $l = 0,2x + \sin(x) + e^{-\sqrt{x \cdot y}}$	Найти произведение и сумму цифр трехзначного числа
10.	$x = 0,5 \frac{\alpha + \beta}{\gamma} - 0,8e^{\alpha + \beta}$ $f = \frac{1,2 \cos(x-0,16)}{0,6 \sin^{1,2} y} + \frac{xy}{\ln x }$	Заданы координаты трех вершин треугольника (x1,y1), (x2,y2), (x3,y3). Найти его периметр и площадь.
11.	$h = \frac{1}{\ln x^{1,25} + y^{2,3} } + \frac{z}{\sqrt[3]{z+y}}$; $g = \frac{x^{1,4} \sqrt{ y } e^{x y }}{\ln x}$	Определить сумму первых двух и последних двух цифр четырехзначного числа.
12.	$f = tg(z^{-2}) - e^{xy}$; $f = z^{xy} + 5 \frac{\sin x+y }{(x+y)^{1,3}}$	Даны основания и высота равнобедренной трапеции. Найти ее периметр и площадь.
13.	$\alpha = \gamma^{1,25} + e^b \frac{x+y}{ x-y }$; $\gamma = b^2 + \cos(x^{1,25}) - \sin y $	Даны длины сторон прямоугольного параллелепипеда. Найти площадь его поверхности.
14.	$h = \sqrt{(\sqrt{ y } + z^2)}$; $s = \frac{tg(y)}{\sqrt[4]{lg(z)}}$	Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.

Окончание таблицы 2

15.	$a = e^{-\sqrt{ b +3\sqrt{ d }}}$; $d = \sqrt[3]{ n c-a} \frac{b^3}{2}$	Определить число, полученное в обратном порядке из заданного трехзначного числа.
-----	--	--

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 3

Составить программу для определения значений логических выражений. Задать значения исходных данных самостоятельно. Вывести на экран значения исходных данных и результата.

Таблица 3 – Работа с логическими переменными

№ п/п	Задание 1	Задание 2
1.	Определить значения логических выражений, утверждающих, что точка с координатами x, y принадлежит окружности радиусом r ; только одна из координат отрицательна.	Записать выражения, истинные при выполнении указанных условий: только одна из логических переменных a и b имеет значение false; обе имеют значение истина.
2.	Определить значения логических выражений, утверждающих, что точка с координатами x, y принадлежит первой или третьей четверти координатной плоскости; хотя бы одно из них больше 10	Написать программу, определяющую значения логических выражений, утверждающих, что два вещественных числа, введённых с клавиатуры, - числа разного знака; только одно из них по модулю меньше 5.
3.	Определить значения логических выражений, утверждающих, что из двух целых чисел a и b , одно чётное, другое нечётное; только одно из них по модулю меньше.	Записать выражения, истинные при выполнении указанных условий: обе переменные a и b имеют четные значения; только одно из них положительно.

Продолжение таблицы 3

№ п/п	Задание 1	Задание 2
4.	Определить значения логических выражений, утверждающих, что три целых числа, введённых с клавиатуры, четные числа; только одно из них отрицательно.	Записать выражения, истинные при выполнении указанных условий: хотя бы одна из логических переменных a и b имеет значение истина; все имеют значение истина.
5.	Записать условия, которые являются истинными, когда: только одно из чисел x , y и z кратно пяти; хотя бы одно из чисел x , y и z больше 100.	Записать условия, которые являются истинными, когда: каждое из чисел X и Y нечетное; только одно из чисел X и Y меньше 20.
6.	Написать программу, которая определяет значения логических выражений, утверждающих, что три целых числа, введённых с клавиатуры, - числа одного знака; хотя бы одно из них меньше 20.	Даны положительные числа x , y и z . Записать условие, которое является истинным, когда $x = \min(x, y, z)$
7.	Записать условия, которые являются истинными, когда: целое N кратно пяти или семи; целое N кратно четырем и не оканчивается нулем.	Записать выражения истинные, если все вещественные числа x , y и z нечетны; хотя бы одно из них больше 0.
8.	Записать условия, которые являются истинными, когда только одно из чисел x , y и z кратно пяти; хотя бы одно из чисел x , y и z больше 10.	Даны три положительных числа x , y , z . Записать условие, которое является истинным, когда $y = \max(x, y, z)$
9.	Даны два четных числа x , y и z – нечетное. Записать условие, которое является истинным, когда $z = \min(x, y, z)$.	Записать условия, которые являются истинными, когда: каждое из чисел A и B больше 100; только одно из них четное.

Окончание таблицы 3

№ п/п	Задание 1	Задание 2
10.	Записать логические выражения, которые имеют значение "истина" только при выполнении указанных условий: $x \geq 0$ и $x < 5$; $x < 3$ или $x \geq 5$.	Записать условия, которые являются истинными, когда: хотя бы одно из чисел X и Y равно нулю; каждое из чисел X , Y и Z отрицательное.
11.	Записать условие, которое является истинным, когда каждое из чисел X и Y отрицательное; только одно из них не кратно 5.	Записать условия, которые являются истинными, когда: каждое из чисел A и B больше 50; только одно из чисел A и B нечетное.
12.	Записать условие, которое является истинным, если хотя бы одно из них отрицательно, и два других числа положительные; каждое из чисел x , y и z кратно трем.	Заданы три стороны треугольника a , b и c . Записать условие, которое является истинным, если этот треугольник является прямоугольным.
13.	Дано трехзначное число. Записать условие, которое является истинным, если число является палиндромом («перевертышем»), то есть таким числом, десятичная запись которого читается одинаково слева направо и справа налево.	Записать условия, которые являются истинными, когда: только одно из чисел A , B и C меньше 5; хотя бы одно из чисел A , B и C отрицательно.
14.	Даны три положительных числа x , y , z . Определить, является ли число y средним арифметическим чисел x и z .	Записать условия, которые являются истинными, когда: ни одно из чисел x , y и z не кратно пяти; хотя бы одно из чисел x , y и z меньше 10.
15.	Записать условия, которые являются истинными, когда: каждое из чисел x и y четное; только одно из чисел не равно 20.	Записать условия, которые являются истинными, когда: только одно из чисел A и B не равно 5; оба кратны 10.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 4

Составить программу для вычисления составной функции. Использовать самостоятельный выбор исходных данных для выполнения всех ветвей алгоритма. При выводе указать какая из ветвей выполняется в данный момент. Решить первое задание с помощью условного оператора (IF) и оператора выбора (CASE), второе задание с помощью только оператора (IF)

Таблица 4 – Разветвляющееся программирование

№ п/п	Задание 1	Задание 2
1.	$F(x) = \begin{cases} \sqrt{x} - 2,7 & \text{если } x > 4 \\ \frac{\sin x}{x^2 + 1}, & \text{если } x < 1 \end{cases}$	Определить и вывести на печать номер квадранта, в котором расположена точка M(x,y).
2.	$F(x) = \begin{cases} x^2 - 4x + 2, & \text{если } x < 1 \\ \frac{\sin(2-x)}{x^2 + 1}, & \text{если } x \geq 2 \end{cases}$	Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны.
3.	$F(x) = \begin{cases} \cos^2(x) - 5^{x+1}, & \text{если } x > 1 \\ \sqrt[5]{\frac{2}{7} + \sqrt{x} + x^7 }, & \text{если } x < 0 \end{cases}$	Написать программу выбора наибольшего из трёх заданных чисел.
4.	$F(x) = \begin{cases} \operatorname{tg}(x) - e^{x+1}, & \text{если } x > 1 \\ 3\sqrt{2 + x^2 + x^3 }, & \text{если } x < 0 \end{cases}$	Найти $\max\{\min(a,b), \min(c,d)\}$
5.	$F(x) = \begin{cases} \ln(y^{-\sqrt{ x }}), & \text{если } x < 0, y = 4,5 \\ \frac{\sin x}{e^{x+y} - 1}, & \text{если } x \geq 1,2 \end{cases}$	Заданы два целых числа. Определить, являются ли они оба четными или оба нечетными или какое из них четное, а какое нечетное.
6.	$F(x) = \begin{cases} 1 - \sin x & \text{если } 5 \leq x < 10, \\ \frac{1}{2}(1 + \cos x), & \text{если } 10 \leq x < 15, \\ \operatorname{ctg}^2 x, & \text{иначе} \end{cases}$	Определить какая сумма цифр заданного четырехзначного числа больше, крайних или средних

№ п/п	Задание 1	Задание 2
7.	$F(x) = \begin{cases} \sqrt[3]{x^2 - 4}, & \text{если } x < 1 \\ \frac{2tg(x+1)}{x^2 - 1}, & \text{если } x \geq 2 \end{cases}$	<p>Определить, принадлежит ли точка М(х,у) кольцу с центром в начале координат, внешним радиусом R1, и внутренним радиусом R2.</p>
8.	$F(x) = \begin{cases} \cos^2(x^2 - 4x + 2), & \text{если } x < 1 \\ \frac{\sin(2-x)}{x^2 + 1}, & \text{если } x \geq 2 \end{cases}$	<p>Даны действительные числа X, Y. Меньшее из этих двух чисел заменить их полусуммой, а большее – их удвоенным произведением.</p>
9.	$F(x) = \begin{cases} x^2 - 4, & \text{если } x < 2,1 \\ \frac{2,5}{x^2 - 1}, & \text{если } x \geq 2,1 \end{cases}$	<p>Даны три числа: a, b и c. Определить, какое из них равно d. Если ни одно не равно d, то найти max (d - a, d - b, d - c).</p>
10.	$F(x) = \begin{cases} \sqrt{e^x + x } + 2x & \text{если } x \leq 0 \\ \frac{2\cos\left(x - \frac{1}{x}\right)}{x^2 + 3x - 4}, & \text{если } x > 3 \end{cases}$	<p>Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник, и если да, то является ли он прямоугольным.</p>
11.	$F(x) = \begin{cases} \lg x^2 - 4, & \text{если } x < 0 \\ \frac{5-x}{\sin x + e^x}, & \text{если } x \geq 1 \end{cases}$	<p>Составить программу, определяющую, пройдет ли график функции $y=ax^2+bx+c$ через заданную точку с координатами (m, n).</p>
12.	$F(x) = \begin{cases} \lg x + e^x, & \text{если } x < 1 \\ \frac{1}{ x^2 - 1 - 3x}, & \text{если } x \geq 1,5 \end{cases}$	<p>Заданы размеры А, В прямоугольного отверстия и размеры х, у, z кирпича. Определить, пройдет ли кирпич через отверстие.</p>

№ п/п	Задание 1	Задание 2
13.	$F(x) = \begin{cases} 0 & \text{если } x \leq 0, \\ x^2 - x & \text{если } 0 < x \leq 1, \\ x^2 - \sin \pi x & \text{для других } x. \end{cases}$	Составить программу для определения подходящего возраста кандидатуры для вступления в брак, используя, следующее соображение: возраст девушки равен половине возраста мужчины плюс 7, возраст мужчины определяется, соответственно, как удвоенный возраст девушки минус 14.
14.	$F(x) = \begin{cases} x^2 - 4, & \text{если } x < 2,1 \\ \frac{2,5}{x^2 - 1}, & \text{если } x \geq 2,1 \end{cases}$	Даны три вещественных числа x , y , z . Вычислить значения выражений $x+y+z$ и $x y z$ и вывести максимальное из них.
15.	$F(x) = \begin{cases} \sin(x) + e^x, & \text{если } x > 2 \\ \sqrt{x^2 + x^3 }, & \text{если } x < 2 \end{cases}$	Определить минимальное число из трех целых чисел.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 5

Составить программы используя, нужный оператор цикла. Исходные данные и результат записать в виде таблицы.

Таблица 5 – Циклическое программирование

№ п/п	Задание № 1	Задание № 2	Задание № 3
1.	Написать программу, которая выводит таблицу квадратов первых десяти целых положительных чисел	Вычислить значение скорости $v = gt - \sqrt{2gh}$, при h , изменяющимся от 12 до 21 с шагом 3. Значение t ввести с клавиатуры, а ускорение свободного падения g принять равным $9,8 \text{ м/с}^2$.	Написать программу, которая вычисляет сумму первых n членов ряда $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ (n задается с клавиатуры)
2.	Найти произведение и сумму всех целых чисел от 8 до 15.	Вычислить значение функции $y = 1 + \sin x^2$, где x – изменяет значения от -1,5 до 6 с шагом 3.	Найти сумму $2 + 2^2 + 2^3 + 2^4 + \dots + 2^n$ без возведения в степень. Число n вводится с клавиатуры
3.	Известны оценки ученика по 10 предметам. Определить среднюю оценку	Вычислить значение функции $y = \sqrt{ x^2 + bx }$, где x – изменяет значения от -3 до 6 с шагом 3.	Составить программу для расчета факториала натурального числа n (факториал числа $n! = 1 \cdot 2 \cdot \dots \cdot n$).

№ п/п	Задание № 1	Задание № 2	Задание № 3
4.	Известны результаты двух спортсменов-пятиборцев в каждом из пяти видов спорта в баллах. Определить сумму баллов, полученных каждым спортсменом.	Работа, совершаемая при подъеме тела вверх по вертикали $A = F_{тяж} \cdot h$. Написать программу вычисления функции A при изменении высоты h от 2 до 8 м с шагом 2.	Дано натуральное число. Определить, является ли сумма его максимальной и минимальной цифр кратной числу a .
5.	Напечатать числа следующим образом: 9 10,4 10 11,4 15 16,4	Плотность воздуха убывает с высотой по закону $p = p_0 e^{-hz}$, где p – плотность на высоте h , $p_0 = 1,29$, $z = 1,25 \cdot 10^{-4}$. Напечатать таблицу зависимости плотности от высоты для значений от 0 до 500 м через каждые 100 м.	При заданном значении x вычислить $\frac{(x-1)(x-3)(x-7)\dots(x-63)}{(x-2)(x-4)(x-8)\dots(x-64)}$
6.	Написать программу, которая вычисляет сумму n целых положительных четных чисел, где число n и значения чисел вводятся с клавиатуры.	Вычислить значение функции $N = \frac{A}{t}$, где A – изменяет значения от 2 до 10 с шагом равным 2. Значение t вводится с клавиатуры.	Написать программу, которая вычисляет сумму первых n нечетных членов ряда $1+3+5+7+\dots+n$

№ п/п	Задание № 1	Задание № 2	Задание № 3
7.	Известен рост каждого ученика двух классов. Определить, в каком классе находится самый низкий ученик. Численность обоих классов одинаковая и вводится с клавиатуры.	Вычислить значение функции где $t = \sqrt{\frac{mk^2}{2}}$, m – изменяет значения от -3 до 6 с шагом 1,5 и k=2.	Дано вещественное число z. Вычислить $S = z + \frac{z^3}{3} + \frac{z^5}{5} + \frac{z^7}{7}$
8.	Найти среднеарифметическое квадратов всех целых чисел от a до b. Значения a и b вводятся с клавиатуры; b > a	Вычислить значение функции $y = 5 \cos x$, где x – изменяет значения от -1,5 до 4,5 с шагом равным 1,5.	Дано вещественное число x. Вычислить $x + \frac{x^2}{3} - \frac{x^4}{5} + \frac{x^6}{7} - \frac{x^8}{9}$
9.	Найти сумму всех целых чисел, меньших 25, которые кратны 2 и 5.	Вычислить значение функции $S = \pi R^2$, где R – изменяет значения от 15 до 25 с шагом равным 5.	Проверить, действительно ли при четном n – целое $P = \frac{n}{2} + \frac{n^2}{4} + \frac{n^3}{8} + \frac{n^4}{16}$
10.	Оценки по информатике восьми учеников вводятся с клавиатуры. Выяснить, какое количество учеников имеют двойки и тройки.	Вычислить значение функции, где d изменяет значения от -6 до 8 с шагом 2, значение b ввести с клавиатуры $x = \frac{b^2 - d}{2}$	Вычислить сумму при x = 1,2 $x + \frac{x^3}{3} + \frac{x^5}{5} + \dots + \frac{x^{11}}{11}$

№ п/п	Задание № 1	Задание № 2	Задание № 3
11.	Найти сумму и произведение всех нечетных чисел от 10 до 20	Вычислить значение функции $W = \frac{m \cdot v^2}{2} + v_0$, где v – изменяет значения от v_1 до v_2 с шагом Δv . Если $m=12$; $v_0=3,2$; $v_1=4$; $v_2=10$ $\Delta v=2$	Вычислить сумму: $S = \frac{2}{3} - \frac{3}{4} + \frac{4}{5} - \frac{5}{6} + \dots + \frac{10}{11}$
12.	Написать программу, которая вычисляет сумму n целых отрицательных чисел кратных 3.	Вычислить значение скорости $V = \frac{S}{t}$, где t – изменяет значения от 3 до 6 с шагом равным 1,5. Значение S ввести с клавиатуры.	Дано натуральное число n . Вычислить $P = \frac{2}{1} + \frac{3}{2} + \frac{4}{3} + \dots + \frac{n+1}{n}$
13.	Составить программу печати таблицы температур по Цельсию от 0^0 до 20^0 и их эквивалентов по шкале Фаренгейта, используя для перевода формулу $t_f = 9t_c/5 + 32$	Вычислить значение давления $P = \frac{F}{S}$, где F – изменяется от 3 до 6 с шагом равным 1,5. Значение S ввести с клавиатуры.	Дано натуральное число n . Вычислить $P = \left(1 - \frac{1}{2^2}\right) \left(1 - \frac{1}{3^2}\right) \dots \left(1 - \frac{1}{n^2}\right)$

№ п/п	Задание № 1	Задание № 2	Задание № 3
14.	Вычислить количество точек с целочисленными координатами, находящихся внутри кольца, радиусом от R1 до R2 > 0	Вычислить значение силы $F = ma - \frac{mv^2}{R}$ при значениях m, изменяющихся от -15 до 30 с шагом 5. Значения ускорения a и скорости движения v ввести с клавиатуры. Значение R принять равным 15 м.	Дано натуральное число n. Вычислить $P = \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{4}\right) \left(1 - \frac{1}{6}\right) \dots \left(1 - \frac{1}{2n}\right)$
15.	Написать программу, которая выводит на экран таблицу умножения на 7	Вычислить значение функции $Q = P \sin \alpha$ при различных значениях аргумента α , изменяющихся от 120 градусов до 210 с шагом 30. Значение P ввести с клавиатуры.	Даны вещественное число a и натуральное число n. Вычислить $S = \frac{1}{a} + \frac{1}{a^2} - \frac{1}{a^4} + \dots + \frac{1}{a^{2n-2}}$

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 6

Составить программы вычисления: значений суммы $S(x)$ и значения функции в диапазоне изменения аргумента x и шагом, указанным в условиях заданий, используя, любые операторы цикла. Результаты оформить в виде таблицы.

Таблица 6 – Циклическое программирование. Цикл в цикле

№ п/п	Задание 1	Задание 2
1.	<p>Вычислить сумму ряда</p> $S = \sum_{k=1}^4 (-1)^k \frac{1+x}{k},$ <p>где x изменяет значения от 2 до 6 с шагом 2.</p>	<p>Вычислить значение функции при различных значениях аргументов: $n=f(d-r)+rs$, где f – изменяется от f_1 до f_2 с шагом Δf; r – изменяется от r_1 до r_2 с шагом Δr. Если $r_1=1800$; $r_2=2700$; $\Delta r=300$; $f_1=12$; $f_2=15$; $\Delta f=1$; $d=1200$</p>
2.	<p>Вычислить сумму ряда</p> $S = \sum_{i=1}^5 (-1)^i \frac{x^i}{i+1},$ <p>где x изменяет значения от 3 до 9 с шагом 3.</p>	<p>Вычислить значение функции при различных значениях аргументов: $w = \frac{\cos(x+y)}{x}$, где x – изменяется от -3 до 3 с шагом 3; y – от -1,5 до 4,5 с шагом 3.</p>
3.	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^3 (-1)^n \frac{(2x)^{2n}}{n},$ <p>где x изменяет значения от 3 до 4 с шагом 0,3.</p>	<p>Вычислить значение функции при различных значениях аргументов: $z = \frac{\cos x}{\sin^2 y}$, где x – изменяется от -2 до 4 с шагом 2; y – от 1 до 2 с шагом 0,5.</p>
4.	<p>Вычислить сумму ряда</p> $S = \sum_{n=0}^5 (-1)^n \frac{x^{n+1}}{2n},$ <p>где x изменяет значения от 1,5 до 4,5 с шагом 1,5.</p>	<p>Вычислить значение функции при различных значениях аргументов: $y = \sqrt{ax^2 + 2}$, где x изменяется от 2 до 6 с шагом 2, а a – меняется от 1 до 2 с шагом 0,5.</p>

№ п/п	Задание 1	Задание 2
5.	<p>Вычислить сумму ряда</p> $S = \sum_{n=0}^5 (-1)^n \frac{x^{2n+1}}{2n-1},$ <p>где x изменяет значения от 1 до 5 с шагом 2.</p>	<p>Вычислить значение функции при различных значениях аргументов $c = \frac{a+b}{\sin \gamma}$, где a – изменяется от -3 до 3 с шагом $\Delta a=3$; γ – изменяется от 1 до 3 с шагом 0,5; b=2</p>
6.	<p>Вычислить сумму ряда</p> $S = \sum_{n=0}^4 (-1)^n \frac{x^{2n}}{2n+1},$ <p>где x изменяет значения от 0,4 до 0,8 с шагом 0,2.</p>	<p>Вычислить значение функции при различных значениях аргументов $y = \sin \frac{a}{t}$, где t – изменяется от -3 до 6 с шагом $\Delta t=3$; a – изменяется от 1,5 до 4,5 с шагом 1,5; k=2</p>
7.	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^3 (-1)^n \frac{\cos n\pi}{n},$ <p>где x изменяет значения от 1 до 5 с шагом 2.</p>	<p>Вычислить значение функции при различных значениях аргументов:</p> $W = \frac{mv^2}{2} + v_0, \text{ где } v_0 = ac + \sin^2 c,$ <p>где m – изменяется от m_1 до m_2 с шагом Δm; v – изменяется от v_1 до v_2 с шагом Δv. Если $m_1=1200$; $m_2=1800$; $\Delta m=100$; $v_1=0,5$; $v_2=1,1$; $\Delta v=0,1$; a=3,1; c=0,9</p>
8.	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^4 \frac{(-1)^n}{(n+1)^2} \left(\frac{x}{3} \right),$ <p>где x изменяет значения от -4,5 до 4,5 с шагом 3.</p>	<p>Вычислить значение функции при различных значениях аргументов: $z = \frac{at^2}{2}$, где a – изменяется от -2 до 4 с шагом 2; t – от -1 до 1 с шагом 0,4.</p>

№ п/п	Задание 1	Задание 2
9.	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^3 (-1)^n \frac{x^{2n}}{2n},$ <p>где x изменяет значения от 2 до 4 с шагом 0,5.</p>	<p>Вычислить значение функции при различных значениях аргументов: $c = \frac{a}{2} + b$, где a – изменяется от -1 до 5 с шагом 3; b – от -1 до 1 с шагом 0,5.</p>
10	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^3 (-1)^n \frac{2n^2 + 1}{2n} x^{2n},$ <p>где x изменяет значения от 2 до 3 с шагом 0,4.</p>	<p>Вычислить значение функции при различных значениях аргументов: $z = \frac{yx}{x+2}$, где x – изменяется от 1 до 2 с шагом 0,4; y – от -0,5 до 1 с шагом 0,5.</p>
11	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^3 (-1)^n \frac{(2x)^{2n}}{2n},$ <p>где x изменяет значения от -3 до 3 с шагом 1,5.</p>	<p>Вычислить значение функции при различных значениях аргументов: $d = \frac{\sin a}{a+b}$, где a – изменяется от -2 до 4 с шагом 2; b – от -1 до 1 с шагом 0,5.</p>
12	<p>Вычислить сумму ряда</p> $S = \sum_{n=0}^3 \frac{x^{2n+1}}{2n+1},$ <p>где x изменяет значения от 2 до 6 с шагом 2.</p>	<p>Вычислить значение функции при различных значениях аргументов: $c = \frac{\sqrt{a^3}}{b+2}$, где a – изменяется от 2 до 3 с шагом 0,3; b – от -2 до 6 с шагом 2.</p>
13	<p>Вычислить сумму ряда</p> $\sum_{n=1}^3 \frac{(-1)^n x^{4n+1}}{2n},$ <p>где x изменяет значения от 1 до 3 с шагом 0,5.</p>	<p>Вычислить значение функции при различных значениях аргументов: $w = \frac{y^3}{\sin x}$, где x – изменяется от -1 до 3 с шагом 0,5; y – от 2 до 4 с шагом 2.</p>

№ п/п	Задание 1	Задание 2
14.	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^3 (-1)^n \frac{\cos x^{2n-1}}{n+2},$ <p>где x изменяет значения от 2 до 3 с шагом 0,3.</p>	<p>Вычислить значение функции при различных значениях аргументов: $W = \frac{x-2y}{y}$, где x – изменяется от -1 до 3 с шагом 0,7; y – от -2 до 4 с шагом 2.</p>
15.	<p>Вычислить сумму ряда</p> $S = \sum_{n=1}^3 (-1)^n \frac{\cos^n x^{\frac{\pi}{4}}}{n+1},$ <p>где x изменяет значения от 2 до 3 с шагом 0,2.</p>	<p>Вычислить значение функции при различных значениях аргументов: $z = \frac{x^3}{3y}$, где x – изменяется от x_1 до x_2 с шагом h; y – от 3 до 9 с шагом 3. Если $x_1 = 1,2$; $x_2 = 2,4$; $y = 0,6$; $h = (x_2 - x_1) / 4$</p>

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 7

Решить с помощью подпрограмм – функции или процедуры. В головной программе произвести ввод исходных данных, вызов подпрограммы и вывод результатов в виде таблицы.

Таблица 7 – Подпрограммы

№ п/п	Задание 1	Задание 2
1.	<p>Вычислить значения функции $y = \frac{\sin x}{1+x^2}$, где x – изменяется от -3 до 3 с шагом равным 1,5.</p>	<p>Вычислить сумму ряда $s = \sum_{k=1}^4 \frac{1+x}{k!}$, где x изменяет значения от 2 до 6 с шагом 2.</p>

№ п/п	Задание 1	Задание 2
2.	Написать программу, которая выводит на экран таблицу значений функции $y= x^n $, где n меняется от 2 до 6 с шагом равным 2.	Вычислить сумму ряда $S = \sum_{n=1}^4 \frac{\ln x}{n!}$, где x изменяет значения от 0,2 до 0,6 с шагом 0,2.
3.	Вычислить значения функции $y = \sqrt{ x^3 }$, где x изменяется от -3 до 4,5 с шагом 1,5.	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{\cos(n+x)}{n!}$, где x изменяет значения от 4 до 12 с шагом 4.
4.	Вычислить значения функции $X = \frac{b^2-d}{2}$, где d изменяется от -2 до 4 с шагом 2, значение b задать константой.	Вычислить сумму ряда $S = \sum_{n=1}^4 \frac{x}{3(n+1)!}$, где x изменяет значения от -4,5 до 4,5 с шагом 3.
5.	Вычислить значения функции $y = 5\cos x$, где x – изменяется от -1,5 до 3 с шагом равным 0,5.	Вычислить сумму ряда $\sum_{n=1}^3 \frac{1}{n!} \left(\frac{x}{3}\right)^{4n}$, где x изменяет значения от -3 до 1,5 с шагом 1,5.
6.	Вычислить значения функции $c = \sqrt{\frac{a}{2} + b}$ при различных значениях аргумента, где a изменяется от 15 до 25 с шагом 5; при $b=0,5$.	Вычислить сумму ряда $S = \sum_{n=1}^4 \frac{(2n+1)}{n!} \cdot x^{2n}$, где x изменяет значения от 3 до 4 с шагом 0,3.
7.	Вычислить значения функции $z = \frac{at^2}{2}$ при различных значениях аргумента, где t	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{2n^2+1}{n!} \cdot x^{2n}$, где x изменяет значения от 2 до 3

№ п/п	Задание 1	Задание 2
	изменяется от 10 до 30 с шагом 10, значение переменной а ввести с клавиатуры.	с шагом 0,4.
8.	Вычислить значения скорости $v = gt - \sqrt{2gh}$ при изменяющемся h от 12 до 21 с шагом 3. Значение t ввести с клавиатуры, а ускорение свободного падения g принять равным 9,8 м/с ² .	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{n^2 + 1}{n!} \cdot \left(\frac{x}{2}\right)^n$, где x изменяет значения от -2 до 4 с шагом 2
9.	Вычислить значения функции при различных значениях аргумента $y = \sin \frac{a}{t}$, где t – изменяется от 30 до 50 с шагом $\Delta t = 10$; значение переменной а ввести с клавиатуры.	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{(2x)^{2n}}{n!}$, где x изменяет значения от -3 до 3 с шагом 1,5.
10.	Вычислить значения функции $S = \pi R^2$, где R – изменяется от 1,5 до 2,5 с шагом равным 0,5.	Вычислить сумму ряда $\sum_{n=1}^3 \frac{x^{4n+1}}{n!}$, где x изменяет значения от 1 до 3 с шагом 0,5.
11.	Вычислить значение функции $N = \frac{A}{t}$, где A – изменяется от 12 до 20 с шагом равным 4. Значение переменной t вводится с клавиатуры.	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{\cos nx}{n!}$, где x изменяет значения от 1 до 5 с шагом 2.
12.	Плотность воздуха убывает с высотой по закону $\rho = \rho_0 \cdot e^{-hz}$, где ρ – плотность на высоте h, $\rho_0 = 1,29$,	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{2n+1}{n!} x^{2n}$, где x изменяет значения от 2 до 3

№ п/п	Задание 1	Задание 2
	$z=1,25 \cdot 10^{-4}$. Напечатать таблицу зависимости плотности от высоты для значений от 0 до 400 м через каждые 100 м	с шагом 0,5.
13.	Работа, совершаемая при подъеме тела вверх по вертикали $A = F_{тяж} \cdot h$. Написать программу вычисления функции A при изменении высоты h от 2 до 4 м с шагом 0,4	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{x^{2n}}{n!}$, где x изменяет значения от 1 до 3 с шагом 0,5.
14.	Вычислить значение функции $y = \sqrt{ x^2 + bx }$, где x – изменяется от -3 до 3 с шагом 1,5.	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{x^n}{n!}$, где x изменяет значения от 3 до 4 с шагом 0,3.
15.	Вычислить значения функции $y = \sqrt{ -4x^2 + 3x + 2 }$, где x изменяется от 0,2 до 0,8 с шагом 0,2	Вычислить сумму ряда $S = \sum_{n=1}^3 \frac{x^n}{(n+1)!}$, где x изменяет значения от 1 до 5 с шагом 2.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 8

Составить программы с применением подпрограмм. Решить задания с помощью любого оператора цикла, используя множественный вариант ввода исходных данных. Записать исходные данные в файл типа real или integer, а промежуточные и результаты – в файл типа text. Воспользоваться формулой для определения расстояния между точками $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Таблица 8 – Файлы

№ п/п	Задание 1	Задание 2
1.	Можно ли построить треугольник? Координаты его вершин вводятся с клавиатуры. Задать множественный вариант ввода координат его вершин. Вычислить количество построенных треугольников.	С клавиатуры в ЭВМ вводятся множество фамилий и рост студентов для определения кандидатов в баскетбольную команду. Вывести на экран фамилии баскетболистов, рост которых больше 170 см.
2.	Определить, являются ли заданные шестизначные числа счастливыми. (Счастливым называют такое шестизначное число, у которого сумма его первых трех цифр равна сумме его последних трех цифр.)	Окружность с центром в начале координат имеет заданный радиус. Вводя последовательно координаты множества точек, являющихся центрами других окружностей того же радиуса R , определить, сколько из этих окружностей пересекает заданную окружность.
3.	Даны множество пар значений x и y . Если x и y отрицательны, то каждое значение заменить его модулем; если отрицательно только одно из них, то оба значения увеличить на 0.5; если оба значения неотрицательны, то оба значения увеличить в 10 раз.	В ЭВМ поступают множества результатов соревнований спортсменов по плаванию. Выбрать и напечатать лучший результат.
4.	В ЭВМ вводятся по очереди координаты множества различных точек. Определить, сколько из них попадет в кольцо с внутренним радиусом R_1 и внешним R_2 .	В продаже книг в книжном магазине принимает участие ЭВМ. Составить программу, которая запрашивает с клавиатуры стоимость книг, сумму денег, внесенную разными

№ п/п	Задание 1	Задание 2
		покупателями, а далее определяет причитающуюся сдачу (если денег внесено больше), печатает «Спасибо», если сдачи не требуется, или выдает сообщение о недостаточности, внесенной суммы.
5.	Определить количество положительных, отрицательных и нулевых значений последовательности целых чисел. Количество членов последовательности заранее неизвестно.	Составить программу, суммирующую различные штрафное время команд (ввод с клавиатуры) при игре в хоккей. Вывести на экран суммарное штрафное время обеих команд. Определить какая из команд заработала меньше штрафного времени.
6.	Даны трехзначные числа. Определить, равен ли квадрат каждого из этих чисел сумме кубов его цифр.	В киоске продается газета стоимостью 300 руб. и журнал стоимостью 800 руб. Составить программу, которая спрашивает о желании различных покупателей (журнал или газета?), принимает деньги (сумма вводится с клавиатуры) и печатает причитающуюся сдачу. Исходные данные задать с клавиатуры.
7.	Даны целые числа. Определить: а) является ли оно четным; б) оканчивается ли оно цифрой 7;	Пассажирский самолет может поднять груз общим весом 30 т. Составить программу для определения веса почтового груза, который можно поместить в самолет

№ п/п	Задание 1	Задание 2
	в) оканчивается ли оно четной цифрой	после посадки пассажиров и загрузки их багажа. Количество пассажиров заранее неизвестно. Во время регистрации пассажиров ЭВМ должна подсчитывать количество пассажиров (условный вес одного пассажира равен 100 кг) и суммировать вес багажа.
8.	Задано множество троек чисел a, b, c . Вводя их по очереди и интерпретируя как длины сторон треугольника, определить, сколько троек может быть использовано для построения треугольника ($a \leq b \leq c$)	В ЭВМ по очереди вводятся фамилии спортсменов и их результаты в соревнованиях по прыжкам в длину. Число участников произвольно. Выдавать на печать лучший результат после выступления очередного спортсмена. После окончания соревнований напечатать итоговое сообщение о победителе.
9.	В ЭВМ вводятся координаты точек. Количество точек заранее неизвестно. Определить, сколько из них попадет в первый квадрат, во второй и между квадратами, если сторона одного квадрата равна a_1 , а другого a_2 и центр квадрата находится в точке с координатами $(0; 0)$.	Составить программу, которая ведет учет очков, набранных каждой командой при игре в баскетбол. После любого изменения счета выводить сообщение. После окончания игры выдать итоговое сообщение. Предусмотреть ввод названий команд в символьных переменных и напечатать их на экране.

№ п/п	Задание 1	Задание 2
10.	Дано множество значений целых чисел m, n . Если числа m и n не равны, то заменить каждое из них числом, равным большему из них, а если равны, то заменить числа нулями.	Вводя в цикле по 3 оценок очередного студента, подсчитать число студентов, не имеющих 2 и 3.
11.	Дано множество значений трехзначных чисел. Выяснить, является ли каждое из них палиндромом («перевертышем»), то есть таким числом, десятичная запись которого читается одинаково слева направо и справа налево	Составить программу, подсчитывающую число удалений в каждой команде при игре в хоккей. После каждого удаления выводить, время, на которое он удален с поля и суммарное число удалений в каждой команде. Определить какая из команд получила большее количество удалений.
12.	Среди чисел $1 < n < 100$ найти все пары чисел, для которых их сумма равна их произведению	Составить программу, которая может выдать необходимые сведения для всех желающих вступить в брак, количество которых заранее неизвестно (например, определение подходящего возраста кандидатуры для вступления в брак и т.д.). Кандидатуры для вступления в брак используют следующее соображение: возраст девушки равен половине возраста мужчины плюс 7, возраст мужчины определяется соответственно как удвоенный воз-

		раст
--	--	------

Окончание т таблицы 8

№ п/п	Задание 1	Задание 2
		девушки минус 14. Исходные данные задать с клавиатуры.
13.	Известны оценки двух учеников по четырем предметам. Определить, какой ученик лучше учится.	Каждый солнечный день улитка, сидящая на дереве, поднимается вверх на 2 см, а каждый пасмурный день опускается вниз на 1 см. В начале наблюдения улитка находилась на расстоянии 20 см от земли на метровом дереве. Написать программу, определяющую местоположение улитки к концу 5-го дня наблюдения.
14.	Даны три различных целых числа. Определить, какое из них (первое, второе или третье): а) самое большое; б) самое маленькое; в) среднее (средним назовем число, которое больше меньшего из данных чисел, но меньше наибольшего).	Составить программу, подсчитывающую число посещений в поликлинике врачей-специалистов (отолоринголога, окулиста, хирурга) в течение дня. Число больных заранее неизвестно. Исходные данные вводятся с клавиатуры. К какому врачу посещений было больше?
15.	Найти все двузначные натуральные числа, которые равны утроенной сумме своих цифр.	Составить программу, подсчитывающую число марок по спорту, по искусству. Подсчитать общее количество марок в коллекции. Исходные данные вводятся с клавиатуры. Количество филателистов заранее неизвестно.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 9

Составить программы с применением подпрограмм, используя одномерные массивы. Для выполнения задания 2 воспользоваться формулой расстояния между точками $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Таблица 9 – Одномерный массив

№ п/п	Задание 1	Задание 2
1.	Создать одномерный массив, содержащий 10 любых чисел. Вычислить сумму всех положительных и сумму всех отрицательных элементов.	На плоскости 5 точек заданы своими координатами: (3;9), (-4; 1), (2;5), (0;-2), (4;-1). Какая из точек наиболее удалена от точки (2;5)?
2.	Создать массив, содержащий 8 целых чисел. Определить и вывести на экран сумму чётных и сумму нечётных элементов.	На плоскости 5 точек заданы своими координатами: (12;-1), (9;-4), (0;1), (-9;24), (3;-13). Все точки соединены с началом координат. Найти длину каждого отрезка и среднюю длину.
3.	Организовать массив, содержащий 12 различных целых чисел. Поменять первые 4 элемента и последние 4 элемента местами.	На плоскости 5 точек заданы своими координатами: (42,1; 0,33), (14,08; 21), (-11; -4), (13,05; -7,1), (21,7; -97,2). Точки, соединены между собой. Определить длину наибольшего отрезка. Найти длину среднего ее звена.
4.	Создать массив, содержащий 8 различных целых чисел. Получить новый массив, умножив все положительные на последний элемент, а отрицательные на первый элемент массива.	На плоскости 4 точек заданы координатами: (-1; 0), (5;4, 1), (0,5; 1), (-6,4; 15). Вычислить сумму расстояний от точки A(13; 8) до остальных точек. Какие точки удалены от точки A на расстояние не менее 4 единиц?

№ п/п	Задание 1	Задание 2
5.	Организуйте массив, содержащий 7 различных символов. Определить сумму положительных и четных элементов, и отрицательных и нечетных.	На плоскости имеется 5 точек с координатами: $(-4,44; 9,1)$, $(-0,13; 5,5)$, $(1,25;7,7)$, $(4;-9,1)$, $(5;-5,1)$. Все точки последовательно соединены между собой. Вычислить среднюю длину звена. Найти длину минимального звена ломаной.
6.	Заполнить массив десятью первыми членами геометрической прогрессии. Задать с клавиатуры первый член прогрессии и знаменатель прогрессии.	На плоскости заданы 5 точек координатами $(4; -1)$, $(5; 18)$, $(-44; 7)$, $(19; 4)$, $(-1; -1)$. Какие точки отстоят от начала координат на расстояние не более 10 единиц?
7.	Написать программу, которая вводит с клавиатуры 8 символов, организывает их хранение в одномерном массиве, а затем выводит содержимое массива в обратном порядке.	На плоскости 5 точек заданы своими координатами: $(0,11; -7,1)$, $(3,21; -17,11)$, $(0,44; 9,1)$, $(-11,3;-7,01)$, $(5;9,11)$. Определить минимальное расстояний от точки до точки и среднее расстояние.
8.	Создайте массив, содержащий 7 различных целых чисел. Получите новый массив, заменив все значения элементов массива противоположными по знаку.	На плоскости заданы 5 точек своими координатами: $(-4,1;2)$, $(3,5;0,1)$, $(2,4;-2,4)$, $(3,5;-2,1)$, $(0,9;4,6)$ и точка $A(7;9)$. Вычислить сумму расстояний от A до остальных точек. Какие точки удалены от A на расстояние не менее 4 единиц?
9.	Создать массив из 6 целых чисел. Преобразовать его следующим образом: все	На плоскости имеется 5 точек с координатами $(14,1; -1,2)$, $(12,7; 0,5)$, $(-20,1; -4,4)$, $(3,57; -11,3)$,

№ п/п	Задание 1	Задание 2
	четные элементы разделить на первый элемент, а нечетные оставить без изменения.	(17,8; -0,7). Все точки последовательно соединены между собой. Вычислить длину замкнутой линии и среднюю длину звена. Найти длину минимального звена ломаной.
10.	Заполнить массив десятью первыми членами арифметической прогрессии. Задать с клавиатуры первый член прогрессии и разность прогрессии.	На плоскости 5 точек заданы своими координатами: (19; 4), (-1; -1), (15; 6), (3,8; 10,1), (3; 4). Какие точки отстоят от начала координат на расстоянии более 6 единиц?
11.	Написать программу, которая вводит с клавиатуры 5 действительных чисел, организовывает их хранение в массиве и создает новый массив следующим образом: к положительным элементам добавляет число 10, от остальных вычитает 10.	На плоскости 5 точек заданы своими координатами: (0,3; 9), (-0,4; 1), (2; 0,5), (0; -0,2), (0,4; 1). Какая из точек наименее удалена от точки (-0,2; 5)? Определить среднее расстояние.
12.	Написать программу, которая вводит с клавиатуры 7 целых чисел, организовывает их хранение в массиве. Затем определяет сумму элементов, значение которых больше среднего арифметического элементов массива.	На плоскости заданы координаты 5 точек: (4; -1), (2; 9), (-11; 0), (17; 7), (-12; -4). Все точки последовательно соединены между собой. Вычислить длину замкнутой линии. Найти длину максимального звена ломаной.
13.	Создать одномерный массив из 10 целых чисел. Определить количество	На плоскости заданы 4 точки своими координатами: (-4; 2), (3; 1), (4; -2), (3; -1) и точка A(4; 7).

№ п/п	Задание 1	Задание 2
	четных и нечетных элементов массива.	Вычислить сумму расстояний от точки А до остальных точек. Какие точки удалены от точки А на расстояние не менее 4 единиц?
14.	Ввести 6 действительных чисел, затем создать новый массив, заменив первые два числа числом 999, а последние два – числом -999. Остальные нулем.	На плоскости 5 точек заданы своими координатами: (1,2; -1), (0,9; 4), (0; 1), (-0,9; 2,4), (3; 1,3). Определить минимальное расстояние от точек до начала координат и сумму всех расстояний.
15.	Написать программу, которая вводит с клавиатуры 7 целых чисел, организует их хранение в одномерном массиве, определяет среднее арифметическое элементов массива.	На плоскости 5 точек заданы своими координатами: (0; -7), (2; -7), (4; 9), (-1; -7), (5; 9). Все точки последовательно соединены между собой. Вычислить длину замкнутой линии и среднюю длину звена. Найти длину максимального звена ломаной.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 10

Составить программы с применением подпрограмм. Записать исходные массивы в текстовый файл, промежуточные данные и результаты дозаписать в тот же файл с пояснением.

Таблица 10 – Одномерные массивы. Запись в файл

№ п/п	Задача 1	Задача 2
1	Заданы массивы А(М), В(Л). Сформировать из них массив, элементы которого равны сумме одноименных	Задан массив А(М). Сформировать новый массив, выбрав только положительные

№ п/п	Задача 1	Задача 2
	элементов массивов, а длина равна длине наименьшего. Упорядочить сформированный массив по возрастанию.	элементы. Удалить минимальный элемент.
2	Заданы массивы $A(M)$, $B(L)$. Сформировать из них массив, элементы которого положительны, найти сумму этих элементов. Упорядочить элементы нового массива по убыванию.	Дан массив целых чисел. Сформировать новый массив, выбрав только четные элементы. Вставить число 999 k -ым элементом.
3	Заданы массивы $A(M)$, $B(L)$. Сформировать из них массив, элементы которого превышают заданное число T . Определить произведение таких элементов. Упорядочить сформированный массив по возрастанию.	Задан массив $B(L)$. Изменить порядок следования элементов исходного массива на обратный и записать в новый массив. Заменить минимальный элемент числом -999 в новом массиве.
4	Заданы массивы $A(M)$, $B(L)$. Сформировать из них массив, элементы которого расположены в обратном порядке элементам исходных массивов. Упорядочить новый массив по убыванию	Дан массив из n трехзначных натуральных чисел. Записать в новый массив только те элементы, у которых первая цифра не равна последней цифре. Удалить r -ый элемент из нового массива.
5	Заданы массивы $A(M)$, $B(L)$. Объединить исходные массивы в один массив путем чередования элементов. Упорядочить полученный массив по убыванию.	Задан массив $A(M)$. Сформировать новый массив из тех элементов, которые не равны заданному числу X . Вставить k -ым элементом число X в новый массив.

№ п/п	Задача 1	Задача 2
6	Заданы массивы $A(M)$, $B(L)$. Сформировать одномерный массив, состоящий из отрицательных элементов массивов, кратных 3 и вычислить произведение таких элементов. Упорядочить новый массив по возрастанию	Задан массив целых чисел. Удалить элементы массива, у которых остаток от деления на 2 равен 3.
7	Заданы массивы. $A(M)$, $B(L)$. Сформировать одномерный массив, состоящий из положительных элементов массивов кратных 3 или 5 и определить сумму таких элементов. Упорядочить полученный массив по убыванию.	В массиве хранятся сведения о количестве осадков, выпавших за каждый день января. Определить: общее количество осадков за месяц; среднеедневное количество осадков. Определить максимальное количество осадков за январь.
8	Заданы массивы $A(M)$, $B(L)$. Сформировать одномерный массив, состоящий из индексов элементов, равных заданному значению X и подсчитать количество таких элементов. Упорядочить массив по возрастанию.	В одномерном массиве целых чисел заменить все элементы, меньшие среднего арифметического, значением среднего арифметического. Удалить минимальный элемент полученного массива.
9	Заданы массивы $A(M)$, $B(L)$. Сформировать одномерный массив, состоящий из отрицательных и нечетных элементов исходных массивов и вычислить сумму таких элементов. Упорядочить полученный массив по убыванию.	Дан массив B состоящий из 12 целых чисел. Поменять местами минимальный элемент с максимальным элементом. Вывести индексы этих элементов.

№ п/п	Задача 1	Задача 2
10	Заданы массивы $A(M)$, $B(L)$. Сформировать одномерный массив, состоящий из положительных и четных элементов исходных массивов и вычислить произведение таких элементов. Упорядочить полученный массив по возрастанию.	На k -е место одномерного массива целых чисел вставить элемент, равный квадрату суммы 3-го и 7-го элементов. Максимальный элемент заменить нулем.
11	Заданы массивы $A(M)$, $B(L)$. Сформировать из них массив, выбрав нули и единицы. Упорядочить полученный массив по возрастанию.	Определить, количество нечетных и отрицательных элементов последовательности целых чисел. Заменить нулями найденные элементы.
12	В одномерном массиве целых чисел заменить все элементы, меньшие среднего арифметического, значением среднего арифметического, округленного до целого. Упорядочить новый массив по возрастанию.	В заданном одномерном массиве поменять местами соседние элементы, стоящие на четных местах с элементами, стоящими на нечетных местах (т.е. 1-й со 2-ым; 3-й с 4-ым и т.д.).
13	Заданы массивы $A(M)$, $B(L)$. Сформировать одномерный массив, состоящий из положительных и четных элементов исходных массивов и вычислить произведение таких элементов. Упорядочить полученный массив по убыванию.	Дан массив размерностью 10. Поменять местами максимальный и минимальный элементы.

№ п/п	Задача 1	Задача 2
14	Заданы упорядоченные массивы $A(M)$, $B(L)$. Сформировать одномерный массив, состоящий из нечетных положительных элементов массива B и четных отрицательных элементов массива A , подсчитать произведение таких элементов. Упорядочить новый массив по убыванию.	Дан массив целых чисел. Все элементы с порядковыми четными номерами увеличить в 2 раза, остальные уменьшить на число a . Заменить минимальный элемент максимальным элементом.
15	Заданы массивы $A(M)$, $B(L)$. Сформировать из них массив, элементы которого равны разности одноименных элементов массивов, а длина равна длине наибольшего. Упорядочить сформированный массив по возрастанию	Задан массив размером 10. Сформировать два массива размером по 5 элементов, включая в первый, элементы с четными индексами, а во второй — с нечетными. Количество четных и нечетных элементов одинаковое.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 11

Составить программы с применением подпрограмм, используя двухмерные массивы. Вывести исходные, промежуточные данные и результаты с пояснениями в текстовый файл.

Таблица 11 – Двухмерные массивы.

№ п/п	Задания 1 – 2
1.	В массиве целых чисел размером 2×4 вычислить сумму всех элементов матрицы.
	Заменить все диагональные элементы матрицы $A(M, M)$ числом 999.
2	Ввести элементы матрицы $A(N, M)$. Заменить элементы матрицы не равные числу K этим числом.

№ п/п	Задания 1 – 2
	Заполнить двухмерный массив (3x4) целыми числами от 1 до 12 по строкам: в первой строке числа от 1 до 3, во 2-ой – от 4 до 6 и т.д. Вычислить сумму первого столбца.
2.	Задана прямоугольная матрица $A(N,M)$. Получить транспонированную матрицу. Создать двухмерную матрицу из одномерного массива, первым столбцом которой являются элементы одномерного массива, а вторым столбцом – их индексы.
3.	Задана квадратная матрица. Заменить отрицательные нечетные элементы матрицы числом P . Создать из квадратной матрицы два одномерных массива, в первом расположены положительные и нулевые элементы, а в другом – отрицательные..
4.	Задана матрица $A(N,M)$. Вычислить разность положительных элементов матрицы. Дана целочисленная матрица. Вычислить сумму элементов k -ой строки. Заменить элементы этой строки числом 999.
5.	Задана квадратная матрица. Вычислить произведение отрицательных нечетных элементов матрицы. Для целочисленной квадратной матрицы в p -ом столбце найти элементы, кратные числу K , подсчитать их количество. Заменить эти элементы нулем.
6	Задана матрица $A(N,M)$. Вычислить сумму нечетных отрицательных элементов матрицы. Дан двухмерный массив целых чисел. В k -ом столбце найти сумму всех нечетных элементов. Заменить все положительные элементы найденной суммой.
7	Дан двухмерный массив целых чисел. Все четные элементы удвоить, а нечетные увеличить на число P . Задана матрица $A(N,K)$. Определить в ней сумму элементов p -го столбца и заменить ее значением элементы k -ой строки.
8	Дан двухмерный массив. Заменить числом 10 все элементы кратные 3 или 5.

№ п/п	Задания 1 – 2
9	Вычислить произведение элементов p -ого столбца квадратной целочисленной матрицы и заменить его значением диагональные элементы.
10	Задана матрица $A(M,N)$. Заменить все нечетные и отрицательные элементы исходной матрицы на их сумму.
	Задана матрица $A(M,M)$. Определить ее минимальный элемент. Разделить на него элементы p -ой строки.
11	Задана матрица $A(M,M)$. Все положительные и кратные 7 элементы матрицы заменить единицами.
	Задана матрица $A(M,M)$. Поменять местами 2-ой и 3-ий столбец матрицы. Определить след исходной матрицы.
12	Заменить все элементы матрицы $A(M,N)$ целых чисел, которые меньше среднего арифметического, квадратами этих элементов.
	Задана прямоугольная матрица $A(M,N)$ Поменять местами 1-й и 2-й строки транспонированной матрицы.
13	Задана матрица $A(N,N)$. Заменить в ней все элементы числом 10.
	Посчитать сумму второго и четвертого столбца элементов матрицы. Заменить ее значением элемента четвертого столбца.
14	Задана матрица $A(M,M)$. Заменить нулями все элементы матрицы не кратные 3 и 5.
	Задана матрица $A(M,M)$. Определить ее минимальный элемент. Разделить на него все элементы матрицы.
15	Заменить все отрицательные элементы двумерного массива A целых чисел квадратами этих элементов.
	Задана матрица. Посчитать количество отрицательных и нечетных элементов в каждом столбце матрицы и записать их в одномерный массив.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 12

Составить программы с применением подпрограмм. Записать в файл типа TEXT исходные данные и дозаписать в файл типа TEXT результат.

Таблица 12 – Двухмерные массивы с записью в файл с применением подпрограмм.

№ п/п	Задания 1 – 2
1.	<p>Дана матрица целых чисел размером 3×4. Определить сумму элементов каждого столбца этой матрицы и записать в одномерный массив. Определить минимальное значение из этих сумм.</p> <p>Задана матрица размером $N \times N$. Удалить k-ую строку. Найти максимальный по модулю элемент новой матрицы.</p>
2.	<p>Все положительные и четные элементы p-ого столбца двухмерного массива, возведенные в квадрат записать в одномерный массив.</p> <p>Заданы две матрицы A и B размером $N \times N$. Сформировать из них прямоугольную матрицу X размером $N \times 2N$, включая в первые N столбцов матрицу A, в следующие — матрицу B.</p>
3.	<p>Даны два одномерных массива одинаковой размерности (6 элементов): один с четными элементами, другой - нечетными. Сформировать матрицу (2×3) из двух строк с чередующимися элементами.</p> <p>Удалить k-ый столбец матрицы. Найти сумму положительных элементов каждой из строк матрицы и записать в одномерный массив.</p>
4.	<p>Дана матрица целых чисел размером 2×3. Определите сумму каждого столбца этой матрицы и заменить p-ый столбец исходной матрицы этой суммой.</p> <p>Удалить k-ю строку матрицы и вычислить сумму каждого столбца матрицы.</p>
5.	<p>Задан одномерный массив. Преобразовать его таким образом, чтобы все четные элементы были равны -1; а нечетные равны 1. Сформировать матрицу из двух строк: первая</p>

№ п/п	Задания 1 – 2
	состоит из -1, а вторая – из 1. Вставить новую строку в исходный массив первой строкой. Вычислить сумму p -го столбца.
6.	Заменить элементы последней строки матрицы суммой элементов каждого столбца. Элементы равные разности между элементами первого и последнего столбцов вставить вторым столбцом.
7.	Дана матрица целых чисел размером 4×3 . Записать четные и положительные элементы k -го столбца в одномерный массив и заменить их нулем в матрице. Дана прямоугольная матрица (3×4). Удалить k -й столбец матрицы. Вычислить сумму диагональных элементов полученной матрицы.
8.	Дана матрица целых чисел размером 4×4 . Заменить элементы k -го столбца, суммой элементов каждой строки. Задана матрица $A(N, M)$, где $N > M$. Получить квадратную матрицу, удалив p -ую строку. Вычислить след полученной матрицы.
9.	Поменять местами k -ю и p -ю строки в матрице. Вычислить произведение элементов p -го столбца. Задана прямоугольная матрица $A(M, N)$. Вставить k -ой строкой сумму элементов второй и третьей строки.
10.	Поменять местами k -й и p -й столбцы. Вычислить произведение элементов p -й строки. Даны вещественные числа a_1, \dots, a_4 и вещественная квадратная матрица размером 4×4 . Получить вещественную матрицу размера 4×5 , вставив в исходную матрицу между третьим и четвертым столбцами новый столбец с элементами a_1, \dots, a_4 .
11.	Дана матрица A символов размером 4×3 . Заменить все максимальные элементы столбцов единицей. Задана матрица $A(N, M)$. Удалить в ней вторую строку. Определить сумму элементов последнего столбца матрицы.

№ п/п	Задания 1 – 2
12.	В заданной матрице заменить p -ю строку и k -го столбец единицами, кроме элемента, расположенного на пересечении p -ой строки и k -го столбца.
	Вставить между вторым и третьим столбец, элементы которого равны числу 999.
13.	Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов. Найти след матрицы.
	Элементы равные сумме элементов первого и последнего столбцов вставить вторым столбцом.
14.	Задана прямоугольная матрица. Поменять местами первый и последний столбцы. Вычислить сумму элементов первой строки.
	Задана квадратная матрица. Удалить из нее столбец, в котором на главной диагонали находится элемент, равный числу k .
15.	Дана матрица целых чисел размером 4×4 . Заменить все элементы последнего столбца суммой элементов последней строки матрицы.
	Элементы равные разности элементов первой и последней строки вставить второй строкой.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 13

Составить программу, используя подпрограммы. Записать в файл типа TEXT исходные данные и дозаписать в файл типа TEXT результат.

Таблица 13 – Работа со строками

№ п/п	Задания 1 – 2
1.	Дана последовательность из n символов. Подсчитать сколько раз среди данных символов встречаются символы $+$, $-$, $*$.
	Во введённой строке удалить все символы, стоящие на нечётных местах.

№ п/п	Задания 1 – 2
2.	В строке, введённой с клавиатуры, удалить все пробелы. Выясните, какая из букв, первая или последняя, встречается во введённой строке чаще?
3.	Дана последовательность из n символов. Выяснить, каких символов в последовательности больше, запятых или точек с запятой. В строке, введённой с клавиатуры, удалить все символы "а", стоящие на четных местах.
4.	Дано слово "ветрянный". Исправить ошибку. Во введённой строке заменить символ '*' символом '!'. Определить позиции символа '*'.
5.	Определить является ли слово, введенное с клавиатуры палиндромом (например, "шалаш", "мадам" и т.д.). Во введённой строке заменить символ "а", стоящий на нечетных местах символом "о".
6.	Дана последовательность из n символов. Выяснить, в какой половине последовательности: первой или второй, больше вопросительных знаков. (Не исключается случай равенства). В строке введённой с клавиатуры заменить все "Х" на "У". Вычислить количество заменяемых символов.
7.	Введённую с клавиатуры строку записать в обратном порядке. Вставить символ "!" после каждого символа пробел.
8.	Ввести строку определенной длины с клавиатуры. Проверить одинаковое ли число открывающих и закрывающих скобок в данной строке Во введённой строке заменить все строчные латинские буквы заглавными. Вставить сочетание символов "!!!!" перед первым символом пробел в строке.
9.	Во введённой строке каждую цифру заменить следующей по порядку цифрой, а цифру девять заменить цифрой ноль. В заданной строке заменить все сочетание "во" символом "?".
10.	Строка содержит произвольный русский текст. Определить каких букв больше: гласных или согласных.

№ п/п	Задания 1 – 2
	В заданной строке после каждой буквы "е" вставить сочетание "!!!".
11.	Дана строка символов. Удалить из нее все рядом стоящие одинаковые символы, оставив по одному. (СССТУУУУУУДДЕННТТТТ - СТУДЕНТ) Дана строка символов. Удалить из нее последний знак препинания.
12.	Дано слово. Если его длина нечетная, то удалить среднюю букву, в противном случае – две средние буквы. Дано предложение. Удалить из него все буквы "о", стоящие на нечетных местах.
13.	Определить длину первого и последнего слова в строке. Определить количество слов в тексте и позиции, с которых они начинаются.
14.	Дана строка символов. Подсчитать количество слов, начинающихся с буквы а. В заданной строке вставить дополнительный пробел между словами.
15.	Дана строка символов. Подсчитать количество букв "к" в последнем ее слове. Дано слово, записываемое через дефис. Поменяйте местами части до и после дефиса и удалите дефис.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 14

Разработать программу с применением подпрограмм, выполняющую ввод списка записей, вывод на экран и в файл в виде таблицы, дозапись в файл выбранных, по заданным условиям, записям. Массив должен содержать не менее 5 записей. Структура записи дана в условии задания.

Таблица 14 – Записи.

№ п/п	Задания 1 – 3
1.	Разработать программу поиска в каталоге, книг, изданных в 1960 году и шифром, начинающихся с буквы "А". Каталог содержит: шифр книги, Ф.И.О. авторов, название книги, год издания.
	Используя записи, написать программу, которая формирует расписание полетов самолетов. Расписание включает в себя номер рейса, пункт назначения, время вылета, тип самолета. Программа выводит номера рейсов, время вылета, типы самолетов всех рейсов на вводимый с клавиатуры пункт назначения.
	Используя записи, написать программу корректировки ведомости, которая содержит порядковый номер; страну производителя, марку машины; стоимость машины в условных единицах. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену марки машины во всех записях; вставку новой записи, в которой вводится новая страна производитель; удаление тех записей, в которых стоимость машины превышает стоимость, заданную с клавиатуры.
2.	Разработать программу, выполняющую поиск товаров, имеющих цену меньше 20тыс. руб. и количеством экземпляров больше 100. Структура записи следующая: шифр товара, наименование товара, цена, количество.
	М учеников проходили тестирование, выполнив 3 теста по какому-либо предмету. Сколько очков набрал каждый ученик по всем темам? Вычислить средний балл, полученный учениками, и разницу между лучшим результатом и средним баллом.
	Написать программу, учитывающую клиентов банка и сохраняющую сведения о том, кто (включая адрес, когда, на какую сумму взял ссуду и что является залогом). Вывести сведения о тех клиентах: <ul style="list-style-type: none"> – которые взяли ссуды выше 300 рублей; – у которых, залогом является автомобиль;

№ п/п	Задания 1 – 3
	вычислить среднюю сумму залога.
3.	<p>Разработать программу, выполняющую поиск всех граждан из списка, родившихся до 1950 года и номером участка = 55. Структура записи следующая: Ф.И.О, домашний адрес, номер участка, год рождения.</p> <p>И спортсменам-многоборцам принимают участие в соревнованиях по М видам спорта. По каждому виду спорта спортсмен набирает определенное количество очков. Вычислить, сколько очков в сумме набрал каждый спортсмен после окончания соревнований, и среднее количество очков. Вычислить разницу в очках для спортсменов, занявших первое и последнее места.</p> <p>Используя записи, написать программу корректировки записей, которая содержит номер ПМК; фамилию начальника ПМК; номер телефона. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену фамилии начальника ПМК по заданному номеру ПМК; вставку новой записи, в которой вводится новое ПМК; удаление записей с номером телефона, заданного с клавиатуры.</p>
4.	<p>Разработать программу, выполняющую поиск всех товаров, имеющихся в наличии и цена которых не превышает 15000 руб. Структура записи следующая: шифр товара, наименование товара, цена (руб.), признак наличия или отсутствия товара.</p> <ul style="list-style-type: none"> – всех сотрудников старше 50 лет; – вычислить средний возраст сотрудников и определить всех, возраст которых ниже среднего. <p>Используя записи, написать программу корректировки записей, которая содержит номер треста; наименование треста; номер СМУ; номер ПМК. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену наименования треста по заданному номеру ПМК; вставку новой записи, в которой вводится новое СМУ; удаление записей с номером СМУ, заданного с клавиатуры.</p>

№ п/п	Задания 1 – 3
	Разработать программу, выполняющую поиск всех студентов, группы "110115", родившихся в 1979 году. Структура записи следующая: ФИО, факультет, группа, год рождения.
5.	Разработать программу, выполняющую поиск книг, названия которых начинаются с буквы "П" и изданных до 1990 года. Структура записи следующая: ФИО авторов, название, год издания, шифр книги.
	Известны данные о сотрудниках фирмы (фамилия, зарплата и пол). Определить: а) фамилию мужчины, имеющего самую большую зарплату (считать, что такой есть и он единственный); б) напечатать данные о сотрудниках, фамилии которых начинается с буквы "А"; в) напечатать данные обо всех сотрудниках, чья зарплата не превышает 300 рублей.
	Используя записи, написать программу корректировки записей, которая содержит номер секции; наименование товара; стоимость товара. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену стоимости товара в одной из секций; вставку новой записи, в которой вводится новая секция; удаление записей с наименованием товара, заданного с клавиатуры.
6.	Разработать программу, выполняющую поиск всех рейсов с номерами большими, чем 50, и вылетающими в "Краснодар". Расписание включает в себя: номер рейса, пункт отправления, пункт назначения, дни полетов.
	Используя записи, написать программу, которая заполняет анкеты студентов. Анкета включает в себя ФИО, возраст, пол, номер группы и оценки по четырем предметам. Программа выводит самого молодого студента.
	Используя записи, написать программу корректировки сведений о том, какие рыбы водятся в Белоруссии. Сведения включают в себя: область; названия озер; наименование рыб. Все данные для корректировки вводятся с клавиатуры.

№ п/п	Задания 1 – 3
	Корректировка включает в себя: замену наименования рыб, вставку новой записи, в которой вводятся данных о новом виде рыб; удаление записей с названием озера, заданного с клавиатуры.
7.	<p>Разработать программу, выполняющую поиск всех граждан, родившихся после 1970 года и фамилии которых начинаются с буквы "П". Структура записи следующая: ФИО, домашний адрес, область, горд или поселок, год рождения.</p> <p>Используя записи, написать программу, которая заполняет анкеты студентов. Анкета включает в себя ФИО, возраст, пол, номер группы и оценки по четырем предметам. Программа выводит ФИО студента, который имеет максимальный балл по четырем предметам.</p> <p>Используя записи, написать программу корректировки ведомости успеваемости, которая содержит номер класса; ФИО классного руководителя; количество учеников; количество учеников, учащихся на 3 и 4; количество учеников, учащихся на 4 и 5. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену фамилии классного руководителя, вставку новой записи, в которой вводятся данных о новом ученике; удаление записей с номером класса, заданного с клавиатуры.</p>
8.	<p>Разработать программу, выполняющую поиск всех номеров групп с максимальным средним баллом по каждой дисциплине и по факультету. Структура записи следующая: номер группы, название дисциплины, средний балл, наименование факультета.</p> <p>Используя записи, написать программу, которая заполняет анкеты студентов. Анкета включает в себя ФИО, возраст, пол, номер группы и оценки по четырем предметам. Программа выводит ФИО студентов мужского пола. Используя записи, написать программу корректировки анкеты, которая включает в себя номер по-порядку; ФИО; год</p>

№ п/п	Задания 1 – 3
	рождения; месяц рождения; число рождения. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену года рождения, фамилия вводится с клавиатуры; вставку новой записи с новой фамилией и всеми остальными данными анкеты; удаление записей, в которой год рождения равен году, введенному с клавиатуры.
9.	<p>Разработать программу, выполняющую поиск всех студентов, с минимальными оценками по каждой дисциплине. Структура записи следующая: ФИО преподавателя, название дисциплины, оценка, ФИО студента.</p> <p>Используя записи, написать программу, которая заполняет анкеты студентов. Анкета включает в себя ФИО, возраст, пол, номер группы и оценки по четырем предметам. Программа выводит средний балл студентов женского пола.</p> <p>Используя записи, написать программу корректировки анкеты, которая включает в себя номер записи, фамилия, наименование предмета, оценка. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену фамилии, которая вводится с клавиатуры; вставку новой записи с новой фамилией и всеми остальными данными анкеты; удаление записей, в которой оценка ниже оценки, введенной с клавиатуры.</p>
10.	<p>Разработать программу, выполняющую поиск записей с температурой свыше 30 градусов в летнее время. Структура записи следующая: номер записи, число от 1 до 31, месяц, температура.</p> <p>Используя записи, написать программу, которая заполняет анкеты студентов. Анкета включает в себя ФИО, возраст, пол, номер группы и оценки по четырем предметам. Программа выводит средний балл студентов мужского пола.</p> <p>Используя записи, написать программу корректировки анкеты, которая включает в себя фамилию автора, название книги, количество страниц и год издания. Все данные для корректировки вводятся с клавиатуры.</p>

№ п/п	Задания 1 – 3
	Корректировка включает в себя: замену фамилии, которая вводится с клавиатуры; вставку новой записи с новой фамилией и всеми остальными данными анкеты; удаление записей, в которой год издания не равен году, введенному с клавиатуры.
11.	<p>Известны фамилии, адреса и телефоны n человек. Найти фамилии и адреса людей, чей телефон начинается с цифры 3. Рассмотреть случаи и напечатать записи, когда телефон задан:</p> <p>а) в виде 7-значного числа (без дефисов); б) с разделяющими дефисами (например, 268-50-59).</p> <p>Используя записи, написать программу, которая формирует расписание движения поездов. Расписание включает в себя номер поезда, пункт назначения, время отправления, тип поезда (пассажирский, скорый), наличие вагона-ресторана. Программа выводит номера поездов, время отправления, тип поезда для всех маршрутов до вводимого с клавиатуры пункта назначения.</p> <p>Используя записи, написать программу корректировки о сотрудниках фирмы, которая содержит фамилию, зарплату и пол. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену женской фамилии, имеющую минимальную зарплату, и повысить ей зарплату на 100 рублей; вставку новой записи, в которой вводится новая фамилия сотрудника, имеющего максимальную зарплату среди мужчин, удаление записей сотрудников, чья зарплата выше средней зарплаты.</p>
12.	Используя записи, написать программу, которая формирует расписание движения поездов. Расписание включает в себя номер поезда, пункт назначения, время отправления, тип поезда (пассажирский, скорый), наличие вагона-ресторана. Программа выводит номера поездов, пункта назначения, время отправления всех скорых поездов/

№ п/п	Задания 1 – 3
	<p>Известны данные о наименовании товаров, стоимости каждого из них, фамилия и адрес поставщика. Составить программу, определяющую, какой из товаров стоит дороже. Распечатать данные, в которых фамилия начинается с буквы "Р", а в адресе есть город "Минск".</p> <p>Используя записи, написать программу, которая формирует расписание движения поездов. Расписание включает в себя номер поезда, пункт назначения, время отправления, тип поезда (пассажирский, скорый), наличие вагона-ресторана. Программа выводит номера поездов, пункта назначения, время отправления всех пассажирских поездов.</p>
13.	<p>Используя записи, написать программу, которая формирует расписание движения поездов. Расписание включает в себя номер поезда, пункт назначения, время отправления, тип поезда (пассажирский, скорый), наличие вагона-ресторана. Программа выводит номера поездов, пункта назначения, время отправления, тип поезда для всех поездов, имеющих вагоны-рестораны.</p> <p>Известны фамилия, должность, вес и пол. Найти:</p> <ul style="list-style-type: none"> а) общую массу мужчин; б) средний рост женщин; в) количество инженеров, в фамилии которых есть буква "Н". <p>Известны данные о сотрудниках фирмы (фамилия, зарплата и пол). Определить:</p> <ul style="list-style-type: none"> а) фамилию мужчины, имеющего самую большую зарплату (считать, что такой есть и он единственный); б) напечатать данные о сотрудниках, фамилии которых начинается с буквы "А"; в) напечатать данные обо всех сотрудниках, чья зарплата не превышает 300 рублей. <p>Известна информация о n клиентах пункта проката: фамилия, адрес и домашний телефон. Известно также название предмета, взятого каждым из них напрокат (в виде t —</p>

№ п/п	Задания 1 – 3
14.	телевизор, х — холодильник, f- фотоаппарат). Вывести на экран фамилию и адрес каждого из клиентов, взявших напрокат телевизор. Распечатать сведения о клиентах, взявших на прокат фотоаппарат, и адрес которых начинается с буквы "А". Вычислить количество клиентов, взявших напрокат холодильник.
	Используя записи, написать программу, которая формирует расписание междугородних автобусов. Расписание включает в себя номер рейса, пункт назначения, время отправления, тип автобуса, количество мест. Программа выводит номера рейсов, пункты назначения, время отправления всех автобусов, имеющих более 20 мест.
	Известна информация о багаже (фамилия пассажира, количество вещей, общий вес багажа) 4 пассажиров. Найти: а) число пассажиров, имеющих более двух вещей; б) хотя бы одного пассажира, багаж которого состоит из одной вещи весом менее 25 кг; в) число пассажиров, у которых количество вещей превосходит среднее число вещей всех пассажиров.
15.	Используя записи, написать программу, которая формирует библиотечный каталог. Каталог включает в себя фамилию автора, название книги, издательство, год, стоимость книги. Программа выводит авторов названия книг, издательство, стоимость всех книг, выпущенных в году, введенном с клавиатуры.
	В телефонной книжке указаны фамилии, номера городского и мобильного телефонов. Составить программу, которая определяет, есть ли в телефонной книжке: а) мобильный телефон человека, фамилия которого вводится с клавиатуры и если есть, напечатать номер этого телефона; б) распечатать данные, чья фамилия начинается с буквы "Д"; в) распечатать и подсчитать количество человек, чьи номера

	телефонов начинаются с "223".
--	-------------------------------

Окончание таблицы 14

№ п/п	Задания 1 – 3
	Используя записи, написать программу корректировки анкеты абонентов, которая включает в себя код абонента; номер телефона абонента; фамилия абонента; адрес абонента. Все данные для корректировки вводятся с клавиатуры. Корректировка включает в себя: замену номера телефона абонента фамилия, которого вводится с клавиатуры, вставку нового абонента; удаление записей с адресом, заданным с клавиатуры.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 15

Составить программу с применением подпрограмм. Исходные данные и результат записать на экран и в файл.

Таблица 15 – Множества

№ п/п	Задания 1 – 2
1.	Вычислить значения выражений: $([2..13] * [3,13..60] + [4..10] - [5..15] * [6])$ Задано множество целых положительных чисел от 1 до n. Создать из элементов этого множества такие подмножества, элементы которых удовлетворяют следующим условиям: не превышают 10; кратны 8; не кратны 3 и 6.
2.	Вычислить значения выражений: $[2..10] - [4,6] - [2..12] * [8..15]$ Строка содержит произвольный русский текст. Проверить, каких букв больше: гласных или согласных.
3.	Вычислить значения выражений: $(['0'..'7'] + ['2'..'9']) * (['a'] + ['z'])$ Дана строка, содержащая текст, заканчивающийся точкой. Вывести на экран и в файл составляющие ее слова из трех букв.
4.	Вычислить значения выражений: $[1,3,5] + [2..4]; [1,3,5] * [2..4]; [1,3,5] - [2..4]$

№ п/п	Задания 1 – 2
	Дан текст, за которым следует точка. Записать на экран и в файл: в алфавитном порядке (по разу) все строчные русские буквы (а, е, и, о, у, ы, э, ю, я), входящие в этот текст.
5.	Записать на экран и в файл в порядке убывания все целые числа из диапазона 1..10000, представимые в виде $n^2 + m^2$, где $n, m \geq 0$ Подсчитать количество разных цифр в десятичной записи натурального числа.
6.	Вычислить значения отношений: $[2] \diamond [2,2,2]$; $['a','b'] = ['b','a']$; $[4,5,6] = [4..6]$; $['c','b'] = ['c','b']$; $[2,3,5,7] \leq [1..9]$. Дан текст на русском языке. Записать на экран и в файл: в алфавитном порядке: все гласные буквы, которые входят в каждое слово; все звонкие согласные буквы, которые входят хотя бы в одно слово.
7.	Вычислить значения отношений: $[3,6..8] \leq [2..7,9]$; $[] \leq ['0'..'9']$; $'q' \text{ in } ['a'..'z']$; $\text{trunc}(3.9) \text{ in } [1,3,5]$; $\text{odd}(4) \text{ in } []$; $[2] < [1..3]$; $66 = [66]$ Вычислить общее количество цифр и знаков '+', '-' и '*', входящих в строку.
8.	Вычислить значение переменной p: $p := [i+3, j \text{ div } 2, j, \text{Sqr}(i) - 3]$; $p := [2*i .. j]$; $p := [1, j, 2*i, 2*j]$. Дан текст на русском языке. Записать на экран и в файл: в алфавитном порядке: все согласные буквы, которые входят только в одно слово; все звонкие согласные буквы, которые входят более чем в одно слово.
9.	Вычислить значения выражений: $[4, 6, 8] + [5, 7]$; $[4, 6, 8] * [5, 7]$; $[4, 6, 8] - [5, 7]$; Пусть дан текст. Найдите наибольшее количество цифр, идущих подряд.
10.	Вычислить значения выражений: $[7 .. 12] + [9 .. 18]$; $[7 .. 12] * [9 .. 18]$; $[7 .. 12] - [9 .. 18]$;

	Пусть дан текст. Верно ли, что в нем имеются буквы, входящие: а) в слово "шина"; б) в слово, задаваемое пользователем?
--	---

Окончание таблицы 15

№ п/п	Задания 1 – 2
11.	Вычислить значения выражений: $[4, 7] + [1..9]$; $[4, 7] * [1..9]$; $[4, 7] - [1..9]$; Пусть дан текст. Определить, содержит ли он символы, отличные от букв и пробелов.
12.	Вычислить значения выражений: $[] + [23]$; $[] * [23]$; $[] - [23]$; Дана строка, содержащая 10 символов. Подсчитать, сколько раз в ней встречается буква "а".
13.	Вычислить значения выражений: $[] + [23]$; $[] * [23]$; $[] - [23]$; Дана строка, содержащая 20 символов. Подсчитать количество цифр, входящих в данную строку.
14.	Вычислить значения отношений $[6] <> [6, 6, 6]$; $['1', '8'] = ['8', '1']$; $[6, 7, 8] = [6..8]$; Дано произвольное слово. Проверить, является ли оно палиндромом. Примеры палиндромов: казак, шалаш, мадам, а также фраза "А роза упала на лапу Азора" и т.д.
15.	Вычислить значения отношений $['n', 'm'] = ['n'..'m']$; $[1, 8] <= [1, 9]$; $[1, 7] <= [1..9]$ Задано множество целых положительных чисел от 1 до n. Создать из элементов этого множества такие подмножества, элементы которых не превышают 10.

ПРИЛОЖЕНИЕ А

Стандартные функции

Обозначение	Тип аргумента	Тип результата	Комментарий
1	2	3	4
Abs(x)	I, R	R, I	x
Sqr(x)	I, R	I, R	x^2
Sin(x)	I, R	R	Sin x, где x аргумент в радианах
Cos(x)	I, R	R	Cos x, где x- аргумент в радианах
Arctan(x)	I, R	R	Arctg x, где x- аргумент в радианах
Exp(x)	I, R	R	e^x
Exp(x*Ln(a))	I, R	R	a^x
Ln(x)	I, R	R	Ln x – натуральный логарифм, где $x > 0$
Ln(x)/Ln(a)			$\text{Log}_a x$
Sin(x)/Cos(x)	I, R	R	Tg x
Cos(x)/Sin(x)	I, R	R	Ctg x
1/Sin(x)	I, R	R	Csc x
1/Cos(x)	I, R	R	Sc x
$\text{Arctan}\left(\sqrt{\frac{x}{1-x^2}}\right)$	I, R	R	Arcsin x
Pi/2- Arcsinx=Pi/2- $\text{Arctan}\left(\sqrt{\frac{x}{1-x^2}}\right)$	I, R	R	Arccos x
Pi/2 –Arctan(x)			Arcctg x
Sqrt(x)	I, R	R	\sqrt{x} , где $x > 0$
Frac(x)	R	R	Дробная часть значения x
Int(x)	R	R	Целая часть значения x
Pi	R	R	Задание числа $\pi = 3.1415926535897932385$

Окончание табл.

1	2	3	4
Pred(x)	I, C	I, C	Предшествующий элемент
Succ(x)	I, C	I, C	Последующий элемент
High(x)	R, I	R, I	Получение максимального значения величины данного типа
Low(x)	R, I	R, I	Получение минимального значения величины данного типа.
Trunc(x)	R	Longint	Вычисление целой части значения x
Round(x)	R	I	Округление числа до ближайшего целого
Chr(x)	I	C	Символ по номеру (коду из таблицы ASCII)
Ord(x)	C	I	Номер по символу (код из таблицы ASCII)
Odd(x)	I	B	True, если x не четное; False, если x не четное
Eof(F)	-	-	Находит конец файла
Eoln(f)	-	-	Находит конец строки в текстовом файле
Random(x)	R	R	Случайное число (0..1)
Random	Word	Word	Случайное число (0..1)
Randomize	Без параметров		Гарантирует несовпадение последовательностей случайных чисел, выдаваемых функцией Random
Inc(x)	I	I	Увеличивает значение x на 1
Dec(x)	I	I	Уменьшает значение x на 1
Inc(x,n)	I	I	Увеличивает значение x на n
Dec(x,n)	I	I	Уменьшает значение x на n

Логические операции

Операция	Пример	Значение А	Значение В	Результат	Название
not	not A	True		False	Логическое отрицание
		False		True	
and	A and B	True	True	True	Логическое И
		True	False	False	
		False	True	False	
		False	False	False	
or	A or B	True	True	True	Логическое ИЛИ
		True	False	True	
		False	True	True	
		False	False	False	
xor	A xor B	True	True	False	Исключающее ИЛИ
		True	False	True	
		False	True	True	
		False	False	False	

ЛИТЕРАТУРА

Основная

1. Фаронов, В.В. Программирование на персональных ЭВМ в среде ТУРБО-ПАСКАЛЬ / В.В. Фаронов. – М.: МГТУ, 1990. – 580 с.
2. Тарануха, Н.А. Обучение программированию: язык Pascal: учебное пособие / Н.А. Тарануха, Л. Гринкруг. – М.: Солон-ПРЕСС, 2009.
3. Окулов, С.М. Основы программирования (Турбо Паскаль) / С.М. Окулов. – 2-е изд. – М.: БИНОМ, 2008.
4. Культин, Н.Б. Самоучитель. Программирование в Turbo Pascal 7.0 и Delphi / Н.Б. Культин. – 3-е изд. – СПб.: БХВ-Петер-бург, 2008. – (Серия: Самоучитель).
5. Вольский, С.В. Turbo Pascal 7.0 для студентов и школьников / С.В. Вольский, П.А. Дмитриев. – М.: Наука и техника, 2007.

Дополнительная

6. Климова, Л.М. Pascal 7.0. Практическое программирование. Решение типовых задач / Л.М. Климова. – 4-е изд. – М.: Кудиц-образ, 2003.
7. Меженный, О.А. Turbo Pascal. Самоучитель / О.А. Меженный. – СПб.: Вильямс, 2008.
8. Фаронов, В.В. Программирование на персональных ЭВМ в среде ТУРБО-ПАСКАЛЬ / В.В. Фаронов. – М.: МГТУ, 1990. – 580 с.

СОДЕРЖАНИЕ

ОСНОВЫ ПРОГРАММИРОВАНИЯ	3
ВВЕДЕНИЕ.....	3
Лабораторная работа № 1	
ИНТЕГРИРОВАННАЯ ИНСТРУМЕНТАЛЬНАЯ ОБОЛОЧКА (ИИО) СИСТЕМЫ TURBO PASCAL 7.0 (ТР 7.0).....	5
Структура основного экрана ИИО ТР 7.0.....	6
Создание программ в ИИО ТР 7.0.....	9
Основы построения программ на ТР 7.0.....	14
Лабораторная работа № 2	
АЛГОРИТМ ЛИНЕЙНОЙ СТРУКТУРЫ.....	17
Константы и переменные.....	18
Комментарий.....	19
Типы данных.....	19
Выражения, операнды, операции.....	24
Операторы.....	27
Программирование линейного алгоритма.....	33
Лабораторная работа № 3	
ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ.....	35
Логический тип данных. Логические выражения и операции..	35
Линейные алгоритмы и программы.....	36
Лабораторная работа № 4	
АЛГОРИТМ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ.....	37
Оператор условного перехода (IF).....	37
Работа оператора.....	38
Оператор выбора (CASE).....	41
Лабораторная работа № 5	
АЛГОРИТМ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ.....	43
Операторы циклов.....	43
Лабораторная работа № 6	
АЛГОРИТМ ЦИКЛ В ЦИКЛЕ.....	53
Организация цикла в цикле.....	54

Лабораторная работа № 7	
ПОДПРОГРАММЫ.....	56
Понятие подпрограммы.....	56
Подпрограмма-процедура.....	56
Подпрограмма-функция.....	59
Лабораторная работа № 8	
ФАЙЛЫ.....	61
Общие сведения о файлах.....	62
Работа с файлами.....	63
Текстовые файлы.....	65
Лабораторная работа № 9	
ОДНОМЕРНЫЕ МАССИВЫ.....	71
Понятие массива.....	71
Операции над массивами и их совместимость.....	72
Лабораторная работа № 10	
ФАЙЛЫ И ОДНОМЕРНЫЕ МАССИВЫ.....	81
Работа с одномерными массивами и файлами.....	81
Лабораторная работа № 11	
ДВУХМЕРНЫЕ МАССИВЫ.....	85
Массивы массивов. Матрицы.....	85
Обработка матриц.....	87
Запись двухмерных массивов в файл и чтение из файла.....	92
Лабораторная работа № 12	
СИМВОЛЬНЫЕ, СТРОКОВЫЕ ПЕРЕМЕННЫЕ.....	93
Символьные переменные. Тип данных "CHAR".....	93
Таблица расширенного кода ASCII.....	94
Функции для работы с символьными переменными.....	96
Символьные массивы. Строки.....	96
Лабораторная работа № 13	
ЗАПИСИ.....	104
Тип данных – записи, их описание и использование.....	104
Оператор присоединения.	
Запись в файл типа данных – запись.....	105
Работа с массивом из записей.....	108

Лабораторная работа № 14

МНОЖЕСТВА	111
Тип "множество"	112
Описание (декларация) типа множество	112
Присвоение для переменных типа множество	113
Операции над множествами	114
ЗАДАНИЯ ПО ПРОГРАММИРОВАНИЮ	117
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 1	117
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 2	119
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 3	121
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 4	124
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 5	127
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 6	132
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 7	135
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 8	138
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 9	144
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 10	147
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 11	151
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 12	154
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 13	156
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 14	158
ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 15	167
ПРИЛОЖЕНИЕ А	170
ПРИЛОЖЕНИЕ Б	172
ЛИТЕРАТУРА	173

Учебное издание

ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0

Лабораторный практикум
по дисциплине «Информатика»
для студентов специальностей

1-70 04 01 «Водохозяйственное строительство»,
1-70 07 01 «Строительство тепловых и атомных станций»,
1-37 03 02 «Кораблестроение и техническая эксплуатация
водного транспорта»

С о с т а в и т е л и:

ДЫТКО Галина Николаевна
СЕНЬКО Ольга Брониславовна

Технический редактор О.В. Дубовик

Подписано в печать 31.05.2010.

Формат 60×84^{1/16}. Бумага офсетная.

Отпечатано на ризографе. Гарнитура Таймс.

Усл. печ. л. 10,29. Уч.-изд. л. 8,04. Тираж 150. Заказ 187.

Издатель и полиграфическое исполнение:

Белорусский национальный технический университет.

ЛИ № 02330/0494349 от 16.03.2009.

Проспект Независимости, 65. 220013, Минск.