Гимназия № 446

Домбровский В.Г.
Лимаренко А.И.

Технология программирования в среде *Turbo Pascal*

Методическое пособие
для 10 – 11 классов
2007

СОДЕРЖАНИЕ.

ВВЕДЕНИЕ.

Turbo Pascal разработан, чтобы удовлетворить требования всех пользователей IBM PS/2, PC и совместимых с ними. Это структурированный язык высокого уровня, который Вы можете использовать для написания программ любого типа и размера.

Turbo Pascal 7.0 построен на основе стандартного Паскаля [1]. Он полностью совместим с кодом, написанным для ранних версий Turbo Pascal. Эта версия так же включает:

- Новую интегрированную среду разработки программ
- Множество накладывающихся окон
- Поддержка мышки, меню, диалоговых окон
- Многофайловый редактор, который может редактировать файлы до 1Мб
- Расширенные возможности отладки
- Полное сохранение и восстановление среды разработки
- Расширенные возможности встроенной справочной системы с возможностью вырезки и вставки кода примеров для каждой библиотечной процедуры и функции.

При создании настоящего методического пособия автор принимал во внимание тот факт, что учащиеся знакомы со следующими темами:

1. Информация и единицы измерения информации.
2. Функциональная схема компьютера.
3. Арифметические основы построения ЭВМ.
4. Логические основы построения ЭВМ.
5. Алгоритм как управляющая информация.
6. Организация памяти компьютера.
7. Массивы, циклы, ветвления.

ГЛАВА 1.

Интегрированная среда программирования Turbo Pascal

Turbo Pascal - это не просто быстрый компилятор текста программы в машинные коды; это эффективный компилятор с интегрированной средой программирования, легкой для изучения и использования (для краткости мы будем называть ее ИСП). С Turbo Pascal не нужно использовать отдельные редактор, компилятор, редактор связей и отладчик для того, чтобы создавать, отлаживать и выполнять свои программы на Паскале. Все эти средства встроены в Turbo Pascal, и все они доступны из ИСП.

Вы можете начать построение своей первой программы Turbo Pascal, используя компилятор, встроенный в ИСП. В конце этой главы Вы изучите среду программирования, напишете и сохраните небольшую программу и изучите некоторые основные навыки программирования.

Встроенная контекстно-ориентированная справочная информация появляется с помощью нажатия клавиш (или отметки мышкой). Можно получить справочную информацию (за исключением случаев, когда управление переходит к Вашей программе) посредством нажатия F1. Меню Help (Alt-H) обеспечивает Вас таблицей содержания системы справочной информации, подробным оглавлением, способностями поиска (Ctrl-F1), возможностью вернуться назад к другим экранам (Alt-F1) и подсказкой по справочной информации (F1, если Вы уже находитесь в системе справочной информации). Любой экран справочной информации может содержать одно или более ключевых слов (высвеченных элементов), по которым можно получить дополнительную справочную информацию.

Компоненты. среды

Можно выделить три видимых компоненты в интегрированной усовершенствованной среде: полоса меню в верхней части, область окна редактора в центре и строка статуса внизу. Многие элементы меню также предлагают диалоговые окна. Прежде чем мы рассмотрим каждый элемент меню в интегрированной среде, давайте опишем эти наиболее общие компоненты.

Полоса меню и подменю.

Полоса меню открывает доступ ко всем командам меню. Полоса меню становится невидимой только в то время, когда Вы просматриваете данные, выводимые вашей программой (Alt-F5). Если полоса меню активна, то заголовок меню будет высвечен; это текущее выбранное меню.

Если за командой меню следует знак многоточия (...), выбор команды приведет к выводу диалогового окна. Если за командой следует стрелка (>), то команда ведет в другое меню. Команда без знака многоточия или без стрелки указывает, что как только Вы ее выбрали, произойдет какое-то действие.

Далее будет показано, как выбрать команды меню, используя только клавиатуру:

1. Нажмите F10. Это делает полосу меню активной.

2. Чтобы выбрать меню, которое Вы хотите посмотреть, используйте клавиши со стрелками. Затем нажмите Enter.

Будет быстрее, если Вы просто нажмете высвеченную букву заголовка меню. Например, из полосы меню нажмите E, чтобы быстро показать Edit меню. Из любого места на-

жмите Alt и высвеченную букву для просмотра требуемого меню.

Чтобы прервать действие, нажмите Esc.

3. Опять используйте клавиши со стрелками для выбора требуемых команд. Затем нажмите Enter.

Опять для быстроты можно нажать высвеченную букву команды для того, чтобы выбрать ее, как только появилось меню.

В этот момент Turbo Pascal выполняет команду или показывает диалоговое окно или показывает другое меню.

Можно также использовать мышку для выбора команд. Процесс заключается в следующем:

1. Отметьте заголовок требуемого меню для его просмотра.

2. Отметьте требуемую команду.

Заметим, что некоторые команды меню являются недоступными, когда нет смысла их выбирать. Вы можете, однако, выбрать (высветить) недоступную команду, чтобы получить по ней подсказку.

Следующая таблица перечисляет наиболее часто используемые Turbo Pascal горячие клавиши.

Таблица 1.1. Общие горячие клавиши.

Клавиша(и) - Элемент меню – Функция
F1 – Help – Показывает экран подсказки.
F2 – File/Save – Сохраняет файл, находящийся в активном окне редактора.
F3 – File/Open – Появляется диалоговое окно и возможность открыть файл.
F4 – Run/Go to Cursor – Запускает Вашу программу до строки, на которой стоит курсор.
F5 – Window/Zoom – Масштабирует активное окно.
F6 – Window/Next – Проходит через все открытые окна.
F7 – Run/Trace Into – Запускает Вашу программу в режиме отладки с заходом внутрь процедур.
F8 – Run/Step Over – Запускает Вашу программу в режиме отладки, минуя вызовы процедур.
F9 – Compile/Make – Делает Make текущего окна F10. Возвращает Вас в полосу меню.

Таблица 1.2. Горячие клавиши меню.

Клавиша(и) - Элемент меню – Функция
Alt-ПРОБЕЛ – Ё меню – Переносит Вас в Ё (System) меню
Alt-C – Compile меню – Переносит Вас в Compile меню
Alt-D – Debug меню – Переносит Вас в Debug меню
Alt-E – Edit меню – Переносит Вас в Edit меню
Alt-F – File меню – Переносит Вас в File меню
Alt-H – Help меню – Переносит Вас в Help меню
Alt-O – Options меню – Переносит Вас в Options меню

Alt-R – Run меню – Переносит Вас в Run меню
Alt-S – Search меню – Переносит Вас в Search меню
Alt-W – Window меню – Переносит Вас в Window меню
Alt-X – File/Exit Завершает Turbo Pascal с выходом в DOS

Таблица 1.3. Горячие клавиши редактирования.

Клавиша(и) - Элемент меню – Функция
Ctrl-Del – Edit/Clear – Удаляет выбранный текст из окна и не помещает его в карман.
Ctrl-Ins – Edit/Copy – Копирует выбранный текст в карман.
Shift-Del – Edit/Cut – Помещает выбранный текст в карман и удаляет его.
Shift-Ins – Edit/Paste – Помещает текст из кармана в активное окно.
Ctrl-L – Search/Search Again – Повторяет последнюю команду Find или Replace.
F2 – File/Save – Сохраняет файл в активном окне редактора.
F3 – File/Open – Позволяет Вам открыть файл.

Таблица 1.4. Горячие клавиши управления окнами.

Клавиша(и) - Элемент меню – Функция
Alt-# - (none) - Показывает окно, где # - номер окна, которое Вы хотите посмотреть.
Alt-0 – Window/List – Показывает список открытых окон.
Alt-F3 – Window/Close – Закрывает активное окно.
Alt-F5 – Window/User Screen – Показывает экран пользователя.
Shift-F6 – Window/Previous – Проходит назад через все открытые окна.
F5 – Window/Zoom – Увеличивает/уменьшает активное окно.
F6 – Window/Next – Проходит вперед через все активные окна.
Ctrl-F5 – Window/Size/Move – Изменяет размер или позицию активного окна.

Таблица 1.5. Горячие клавиши встроенной справочной информации.

Клавиша(и) - Элемент меню – Функция
F1 – Help/Contents – Открывает контекстно-ориентированный экран справочной информации.
F1 F1 – Help/Help on Help – Вызывает справочную информацию по справочной информации (нужно нажать только F1, если Вы уже находитесь в системе справочной информации).
Shift-F1 – Help/Index – Вызывает оглавление справочной информации.
Alt-F1 – Help/Previous Topic – Показывает предыдущий экран справочной информации.
Ctrl-F1 – Help/Topic Search – Вызывает специфическую информацию по языку только в редакторе.

Таблица 1.6. Горячие клавиши отладки/запуска.

Клавиша(и) - Элемент меню – Функция

Alt-F9 – Compile/Compile – Компилирует последний файл в редакторе.
Ctrl-F2 – Run/Program Reset – Переустанавливает выполняемую программу.
Ctrl-F4 – Debug/Evaluate/Modify – Вычисляет выражение.
Ctrl-F7 – Debug/Add Watch – Добавляет выражение для просмотра.
Ctrl-F8 – Debug/Toggle BreakPoint – Устанавливает или очищает условные точки прерывания.
Ctrl-F9 – Run/Run – Запускает программу.
F4 – Run/Go To Cursor – Запускает программу до позиции курсора.
F7 – Run/Trace Into – Выполняет прослеживание внутри процедур.
F8 – Run/Step Over – Осуществляет перескакивание через вызовы процедур.
F9 – Compile/Make – Выполняет Make (компилирует/ редактирует связи) программы.

Окна Turbo Pascal.

Почти все, что Вы видите и делаете в среде Turbo Pascal, происходит в окнах. Окно - это область экрана, которую можно перемещать, изменять ее размеры, перекрывать, закрывать и открывать.

Вы можете иметь любое количество открытых окон в Turbo Pascal (если память позволяет), но в любой момент времени может быть активным только одно окно. Активное окно - это окно, с которым Вы в настоящий момент времени работаете. Любая команда, которую Вы выбрали или текст, который Вы набрали, относится только к активному окну. (Если один и тот же файл открыт в нескольких окнах, действие будет применяться к файлу везде).

Существуют несколько типов окон, но большинство из них имеют несколько общих областей:

- полоса заголовка;
- закрывающая кнопка;
- полосы прокрутки;
- уголок для изменения размеров окна;
- кнопка масштабирования;
- номер окна.

В Turbo Pascal легко отличить активное окно по двойной рамочке. Активное окно всегда имеет закрывающую кнопку, кнопку масштабирования, кнопки перемещения и уголок изменения размеров. Если Ваши окна перекрываются, то активное окно всегда находится наверху остальных (на переднем плане).

Окно редактора всегда показывает номера текущих строки и столбца в нижнем левом углу. Если вы изменили свой файл, то слева от номеров строки и столбца появится знак звездочки (*).

Закрывающая кнопка окна - это кнопка в левом верхнем углу. Отметив эту кнопку, Вы можете быстро закрыть окно. (Можно также выбрать Window/Close или нажать Alt-F3). Окно справочной информации рассматривается как временное и может быть закрыто посредством нажатия Esc.

Полоса заголовка - находящаяся выше всех горизонтальная строка окна, содержащая имя окна и номер окна. Вы можете дважды отметить кнопку, находясь на полосе заголовка, для того чтобы масштабировать окно. Вы можете также тащить за строку заголовка для

перемещения окна.

Каждое открытое окно в Turbo Pascal имеет номер в верхнем правом углу. Alt-0 выдает список всех открытых окон. Можно сделать окно активным (самым верхним) посредством нажатия Alt в комбинации с номером окна. Например, если окно справочной информации является #5, но скрыто под другими окнами, то для быстрого вынесения его на передний план можно нажать Alt-5.

Кнопка масштабирования окна находится в верхнем правом углу окна. Если значок в этом углу изображает стрелку вверх, можно отметить эту стрелку для увеличения окна до максимально возможного размера. Если значок представляет собой двуглавую стрелку, то окно уже имеет свой максимальный размер. Если Вы отметите двуглавую стрелку, то окно вернется к своему предыдущему размеру. Чтобы масштабировать окно с помощью клавиатуры, выберите Window/Zoom или нажмите F5.

Полосы скроллинга (прокрутки) - это вертикальные или горизонтальные полосы. Вы можете использовать эти полосы с помощью мышки для перемещения содержимого окна. Отметьте стрелку на любом конце для перемещения одной строчки. (Для непрерывного перемещения держите кнопку мыши нажатой). Отмечая затененную область с любой стороны полосы скроллинга, Вы можете перескочить на страницу. Наконец, можно тащить кнопку прокрутки "за любое место" для быстрого перемещения в окне в место, соответствующее позиции полосы прокрутки.

Примечание. Полосы прокрутки позволяют пользователям видеть, в каком месте файла они находятся.

Уголок изменения размеров находится в нижнем правом углу окна. Можно тащить за любой уголок, чтобы сделать окно больше или меньше. Вы можете отличить уголок изменения размеров по рамке из одной линии вместо рамки из двойной линии, используемой в остальных местах окна. Для того чтобы изменить размеры с помощью клавиатуры, выберите Size/Move из Window меню, или нажмите Ctrl-F5.

Управление окнами.

Таблица 1.7 содержит краткую информацию о том, как можно управлять окнами в Turbo Pascal. Заметим, что для выполнения этих действий мышка не нужна - клавиатура работает прекрасно.

Таблица 1.7. Управление окнами.

Чтобы выполнить:	Используйте один из этих способов
Открыть Edit окно:	Выберите File/Open для открытия файла и покажите его в окне или нажмите F3.
Открыть другие окна:	Выберите требуемое окно из Window меню.
Закрыть окно:	Выберите Close из Window меню (или нажмите Alt-F3), или отметьте закрывающую кнопку окна.
Активизировать окно:	Щелкните в любом месте окна, или нажмите Alt плюс номер окна (в верхнем правом углу окна), или выберите Window/List или нажмите Alt-0 и выберите окно из списка, или выберите Window/Next или F6,чтобы сделать активным следующее окно (следующее в порядке, в каком они были открыты) или нажмите Alt-F6 для активизации предыдущего окна.

Передвинуть: Тащите его заголовок, или нажмите Ctrl-F5 активное окно (Window/Size/Move) и используйте клавиши со стрелками, чтобы поместить окно там, где Вам хочется, и нажмите Enter.

Изменить размер: Тащите уголок для изменения размера (или любой активного окна угол). или выберите Window/Size/Move и нажимайте Shift одновременно с клавишами со стрелками, и нажмите Enter. Более быстрый способ состоит в нажатии Ctrl-F5 и затем одновременным использованием Shift и клавиш со стрелками.

Масштабировать: Отметьте кнопку масштабирования в верхнем активное окно правом углу окна, или дважды щелкните по заголовку окна, или выберите Window/Zoom, или нажмите F5.

Строка статуса.

Строка статуса появляется внизу экрана Turbo Pascal. Функции строки статуса следующие:

- Она напоминает Вам назначение основных горячих клавиш, допустимых в этот момент в активном окне.
- Она предоставляет Вам самый быстрый вариант выполнения действий, отмечая горячие клавиши в строке статуса мышкой вместо выбора команд из меню или нажатия последовательности клавишей.
- Она говорит о том, какая функция выполняется. Например, она показывает "Saving filename...", когда сохраняется редактируемый файл.
- Она предлагает краткие советы по выбранной команде меню и элементам диалогового окна.

Как только Вы переключили окна или изменили характер деятельности, строка статуса сразу же меняется. Одна из наиболее характерных строк статуса - это та, которую Вы видите во время написания и редактирования программ в окне редактора. Она выглядит следующим образом:

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Диалоговые окна.

Если после команды меню следует многоточие (...), команда открывает диалоговое окно. Диалоговое окно - это наиболее удобный способ показать и установить многочисленные опции (настройки).

Когда Вы делаете установки в диалоговых окнах, Вы работаете с пятью основными способами управления экраном: зависимые кнопки, независимые кнопки, кнопки действия, окна ввода и окна списка.

Диалоговое окно имеет три стандартные кнопки: OK, Cancel, Help. При нажатии OK будут выполнены все выборы в диалоговом окне; при выборе Cancel никаких изменений и действий сделано не будет, а диалоговое окно исчезнет. Esc всегда является быстрым вариантом клавиатуры для Cancel (даже если Cancel кнопка не присутствует).

Если Вы используете мышку, Вы можете только отметить требуемую кнопку. Когда Вы используете клавиатуру, Вы можете нажимать высвеченную букву элемента для его активизации. Например, нажатие K означает выбор кнопки OK. Нажимайте Tab или Shift-

Tab для перехода от одного элемента к другому в диалоговом окне. Каждый элемент, когда он становится активным, высвечивается.

В диалоговом окне OK является кнопкой, заданной по умолчанию; это означает, что достаточно нажать только Enter для выбора этой кнопки. Помните о том, что переход к кнопке с помощью табуляции делает ее кнопкой по умолчанию.

Окна ввода и списки.

Возможно, Вы уже знакомы с окнами ввода; эти окна позволяют Вам набирать текст. В окне ввода работают самые основные клавиши редактирования текста (например, клавиши со стрелками, Home, End и переключение режима вставки/перекрытия с помощью Ins). Если, достигнув конца окна, Вы продолжаете набирать текст, он автоматически передвигается. Если текста больше, чем видно в окне, то на концах появляются значки стрелок (влево или вправо). Можно щелкать по этим значкам мышкой для скроллинга текста.

Редактирование.

Если Вы давно пользуетесь продукцией фирмы Borland, то следующее краткое изложение новых свойств редактора поможет вам узнать те области, где произошли изменения.

Интегрированный редактор Turbo Pascal теперь имеет:

- поддержку мышки;
- поддержку больших файлов (до 1 Мб; ограничение в 2 мегабайт для всех комбинаций редактора);
- Shift и стрелки - для выбора текста;
- окна редактора, которые можно передвигать, перекрывать и изменять их размеры;
- мультифайловые возможности, что позволяет открывать несколько файлов одновременно;
- многочисленные окна, позволяющие иметь несколько представлений одного и того же файла или разных файлов;
- возможность брать текст или примеры из окна справочной информации;
- редактируемый карман, допускающий вырезание, копирование и его передачу между окнами.

Запуск Turbo Pascal.

Если Вы используете устройство с гибкими дисками, то поместите диск с системой Turbo Pascal на устройство A и наберите следующую команду:

```
TURBO
```

Нажмите Enter для запуска программы TURBO.EXE, которая вызовет ИСП.

Если Вы используете жесткий диск, то перейдите в каталог Turbo Pascal (по умолчанию C:\TP70) и запустите TURBO.EXE, набрав:

```
TURBO
```

в ответ на подсказку C:\TP. Теперь Вы готовы к написанию своей первой программы Turbo Pascal.

Создание первой программы.

При загрузке Turbo Pascal (набрав TURBO и нажав Enter в ответ на подсказку DOS), Вы увидите полосу меню, строку статуса, пустой экран и окно с информацией о версии продукта (выбор команды About (о) из меню Ё, или System (системного), в любой момент времени приведет к появлению этой информации). При нажатии любой клавиши информация с версией исчезает, но среда с окнами остается.

Нажмите F10, чтобы войти в меню, а затем F3 (сокращение для File/Open) для показа диалогового окна Open a File. Вы находитесь в окне ввода, поэтому наберите MYFIRST (необязательно набирать расширение .PAS; это предполагается и так) и нажмите Enter. Теперь можно приступить к набору текста программы, нажимая Enter в конце каждой строки:

```
program MyFirst;
  var
    A,B: Integer;
    Ratio: Real;
begin
  Write('Enter two numbers: ');
  Readln(A,B);
  Ratio := A/B;
  Writeln('The ratio is ',Ratio);
  Write('Press <Enter>...');
  Readln;
end.
```

Примечание. Не забывайте про точку с запятой, а за последним end поставьте точку.

Для удаления используйте Backspace, а для передвижения внутри окна редактора используйте клавиши со стрелками.

Анализ первой программы.

Поскольку Вы набираете и запускаете эту программу, не зная, что она делает, мы приведем здесь точное пояснение. Первая введенная строка задает имя программе MyFirst. Это утверждение необязательно, но неплохо включать его в программу.

Следующие три строки объявляют несколько переменных, со словом var, говорящем о начале объявления переменных. A и B имеют тип Integer (целое число); то есть, они могут содержать целые числа, такие как 52, -421, 0, 32, 283 и так далее. Ratio объявлено как Real (действительное число), это означает, что оно может хранить дробные числа, такие как 423.328 и -0.032, в добавление к целым числам (обратите внимание, что десятичная запятая заменена на десятичную точку).

Остальная часть программы содержит выполняемые утверждения. Слово begin говорит о начале программы. Утверждения (операторы) разделяются точкой с запятой и содержат инструкции записи на экран (Write и Writeln), чтения с клавиатуры (Readln) и выполнения

вычислений (Ratio := A/B). Readln в конце программы приводит к остановке выполнения (пока Вы не нажмете Enter), так что Вы сможете исследовать на экране выводимые программой данные. Выполнение программы начинается с первой инструкции после begin и продолжается до тех пор, пока не встретится end.

Сохранение первой программы.

После ввода Вашей первой программы было бы неплохо сохранить ее на диске. Для этого выберите команду Save из меню File, нажав F10, затем F для появления меню File и S для выбора команды Save. Наиболее легкий способ - используйте сокращение для File/Save, F2.

Компиляция первой программы.

Чтобы процессор компьютера смог выполнить программу, записанную на языке высокого уровня, необходимо поставить в соответствие каждому оператору Паскаля ряд машинных команд, позволяющих осуществить указанное в операторе действие. Этим занимается специальная программа, входящая в состав ИСП – компилятор.

Компилятор – это переводчик текста программы с языка высокого уровня, понятного человеку на язык машинных кодов, понятный процессору компьютера.

Компилятор временно создает в памяти компьютера или на диске исполняемую программу в машинных кодах. Если записать эту программу на диск в виде файла, ей присваивается расширение EXE и ее можно использовать автономно, без ИСП Turbo Pascal (поговорим об этом позднее).

Для компиляции своей первой программы нажмите Alt-F9 - это самый быстрый путь (можно также воспользоваться меню)..

Turbo Pascal компилирует Вашу программу, переводя ее с Паскаля (который можно читать) на машинный код 8086 для микропроцессора (который может выполнить Ваш компьютер). Вы не увидите машинный код 8086; он хранится в памяти (или на диске).

Как и английский язык, Паскаль имеет грамматические правила, которые нужно выполнять. Однако, в отличие от английского языка, структура языка Паскаль является нетерпимой к сленгу или неправильному синтаксису - компилятор должен всегда понимать, ЧТО Вы хотите сказать. В языке Паскаль, если Вы используете несоответствующие слова или символы в утверждении или если неправильно его организовали, то получите ошибку времени компиляции (синтаксическую).

Какие ошибки компиляции вероятнее всего Вы получите? Наиболее возможной ошибкой для начинающего программиста на языке Паскаль будет:

Unknown identifier (неизвестный идентификатор)

или

';' expected (ожидается ';')

Паскаль требует, чтобы Вы объявили все переменные, типы данных, константы и подпрограммы - короче, все идентификаторы - перед их использованием. Если Вы обратитесь к необъявленному идентификатору или пропустите его, то получите ошибку.

Другой частой ошибкой является несоответствие пар begin..end, присваивание несовместимым типам данных (например, присваивание вещественного числа целой переменной), число и тип параметров не соответствуют в вызовах процедур и функций и т.д.

Когда Вы начинаете компиляцию, в центре экрана появляется окно, содержащее информацию о данной компиляции. Если во время компиляции не произошло никаких ошибок, то в этом окне появится сообщение "Compilation successful: press any key" (компиляция успешна: нажмите любую клавишу). Окно остается на экране до тех пор, пока Вы не нажмете клавишу.

Если во время компиляции произошла ошибка, Turbo Pascal останавливает компиляцию, устанавливает курсор на ошибку в окне редактора и показывает сообщение об ошибке вверху окна редактора. (Первое нажатие клавиши очистит это сообщение, а Ctrl-Q, W будет показывать его снова до тех пор, пока Вы не измените файл или не перекомпилируете его). Сделайте исправления, сохраните обновленный файл и компилируйте снова.

Выполнение первой программы.

После фиксации ошибок идите в основное меню и выберите Run/Run (или нажмите Ctrl-F9). Вы находитесь на экране пользователя, и на этом экране появится сообщение (только для нашей первой программы !):

Enter two numbers:

Наберите два любых целых числа с пробелом между ними и нажмите Enter. Появится следующее сообщение:

The ratio is

а за ним - отношение первого числа ко второму. В следующей строке появится сообщение "Press <Enter>...", а программа будет ждать нажатия клавиши Enter. Чтобы посмотреть вывод своей программы, выберите Window/User Screen (или нажмите Alt-F5).

Если во время выполнения программы произошла ошибка, то на экране появится сообщение, которое выглядит следующим образом:

Run-time error <error number> at :<offset>

где <error number> - это соответствующий номер ошибки (см. в приложении "Сообщения об ошибках" информацию по ошибкам компиляции и ошибкам времени выполнения), а :<offset> - это адрес в памяти, где произошла ошибка. (Если Вы хотите вернуться к этой ошибке позднее, ищите ее в окне Output). Вы окажетесь в точке расположения ошибки в своей программе с описательным сообщением об ошибке, показанным в строке статуса редактора. Пока сообщение находится в строке статуса редактора, можно нажать F1 для получения справочной информации по конкретной ошибке. Нажатие любой другой клавиши приводит к исчезновению сообщения об ошибке. Если Вам нужно будет найти местоположение ошибки снова, выберите Search/Find Error.

Когда Ваша программа закончит выполнение, Вы вернетесь в то место программы, с которого начинали. Теперь Вы можете модифицировать программу, если хотите. Если Вы выберете команду Run/Run перед внесением изменений в свою программу, Turbo Pascal

немедленно выполнит ее снова без перекомпиляции.

Как только Вы вернетесь назад в ИСП после выполнения своей программы, Вы можете просмотреть вывод своей программы посредством выбора команды Run/User Screen (или нажатия Alt-F5). Выберите ее снова для возврата в среду Turbo Pascal.

Проверка файлов, которые Вы создали.

Если Вы вышли из Turbo Pascal (Alt-X или выбрав Exit из меню File), Вы можете просмотреть исходный файл (Паскаль), который Вы создали. Для этого в Norton Commander укажите курсором на ваш файл MYFIRST.PAS и нажмите F3. Чтобы выйти после просмотра, нажмите Esc.

Файл MYFIRST.PAS содержит программу на Паскале, которую Вы только что написали. Если Вы сохраняли программу во время ее написания, Вы также увидите файл MYFIRST.BAK, который был автоматически создан редактором.

Вы увидите выполнимый файл только в том случае, если Вы изменили установку по умолчанию Destination в меню Compile на Disk. Вы можете затем создать файл, названный MYFIRST.EXE, который будет содержать машинный код, сгенерированный Turbo Pascal из Вашей программы. Затем Вы сможете выполнить эту программу, набрав MYFIRST, а затем нажав Enter в ответ на системную подсказку DOS без запуска Turbo Pascal.

ГЛАВА 2.

ПРОГРАММИРОВАНИЕ НА TURBO PASCAL.

Язык Паскаль был разработан Виртом в начале 70-х годов, как язык обучения программированию. Если у Вас есть минимальный опыт программирования, то Вам будет не трудно его освоить (вспомните среду программирования КУМИР).

Элементы программирования.

Большинство программ создаются для решения какой-нибудь задачи. Решение задачи достигается обработкой информации или данных. Поэтому, как программисту, Вам необходимо знать, как:

- предоставить необходимую для работы программы информацию (ввод);
- сохранять информацию (данные);
- задать правильные команды обработки данных (операции);
- получить данные из программы (вывод).

Вы можете написать и упорядочить свою программу так, чтобы:

- некоторые из операторов выполнялись при истинности условия или ряда условий (условное выполнение или ветвление);
- некоторые операторы повторялись определенное число раз (циклы);
- некоторые операторы объединялись в отдельные подпрограммы, которые могут быть выполнены в нескольких местах программы (процедуры).

Основные понятий программирования: ввод и вывод данных, операторы, ветвление, циклы и процедуры. Этот список неполный, но он содержит основные понятия, присущие всем программам.

Многие языки программирования, включая Паскаль, имеют свои особенности. Но когда Вы хотите быстро изучить новый язык, Вы можете посмотреть как он реализует эти семь элементов и начать работать. Дадим краткое описание каждого элемента.

Ввод – это процесс, при котором информация поступает с клавиатуры, диска или порта ввода/вывода в память компьютера.

Данные – это информация, которая хранится в константах, переменных и структурах, содержащих числа (целые и вещественные), текст (символы и строки).

Операторы – команды, осуществляющие присваивание значений, вычисление выражений (сложение, деление и т.д.), сравнение значений (равно, не равно, больше) или выполняют другие действия. Операторы записываются в виде слова или небольшого предложения на английском языке по определенным правилам, чтобы компилятор смог их расшифровать.

Вывод - это процесс, при котором информация из памяти поступает на экран компьютера, записывается на диск или в порт ввода/вывода.

Условное выполнение (ветвление) – это выполнение одного или нескольких операторов, если истинно некоторое условие (если условие ложно, то эти операторы пропускаются или выполняется другой набор операторов). Иногда используется множественное ветвление, если элемент данных имеет указанное значение или диапазон значений.

Циклы – конструкции, в которых набор операторов выполняется определенное число раз, или пока истинно некоторое условие, или пока условие не станет истинным.

Процедуры - набор операторов, объединенных под одним именем, которые выполняются в любом месте программы, где есть вызов процедуры по имени.

Рассмотрим, как использовать эти элементы в Turbo Pascal.

Типы данных.

В процессе программирования программист имеет дело с информацией, представляющей из себя пять основных типов:

- целые числа,
- вещественные числа,
- символы,
- строки символов,
- булевские данные.

Целые числа - это натуральные числа, с помощью которых Вы учились считать (1, 5, -21 и 752, например).

Вещественные числа – это дробные числа (3.14159) и экспоненты (2.579x10**24). Они также известны как числа с плавающей точкой.

Символы - это любые буквы алфавита, символы и цифры 0-9. Они могут использоваться отдельно (a, z, !, 3) или объединяться в символьные строки, заключаемые в апострофы ('Это только проверка').

Булевские выражения имеют только два значения: TRUE или FALSE (истина или ложь). Они используются в условных выражениях и при выполнении ветвления.

Целые числа.

В стандартном Паскале целочисленный тип определяется в пределах от –MaxInt до MaxInt, где MaxInt - наибольшее возможное целое значение, допустимое для процессора.

В Turbo Pascal поддерживается целочисленный тип MaxInt = 32,767, допуская значение -32,768. Переменная целочисленного типа занимает 2 байта.

В Turbo Pascal, кроме того, поддерживается четыре других целочисленных типов данных, каждый из которых имеет свой диапазон значений.

Таблица 2.1. Целочисленные типы данных.

Тип	Диапазон	Размер в байтах
Byte	0..255	1 б (целое длиной в байт)
Shortint	-128..127	1 б (короткое целое)
Integer	-32768..32767	2 б (целое)
Word	0..65535	2 б (длиной в слово)
Longint	-2147483648..2147483647	4 б (длинное целое)

Turbo Pascal позволяет использовать шестнадцатеричные целые значения (основание 16). При описании шестнадцатеричной константы перед ней указывается знак доллара $, например $27=39.

Вещественный тип данных.

В стандартном Паскале тип Real представляет значение с плавающей точкой, содержащее мантиссу и экспоненту - степень числа 10.

Количество значащих цифр в мантиссе и диапазон значений экспоненты зависят от компьютера. В Turbo Pascal данные вещественного типа имеют размер в 6 байт, с 11 значащими цифрами мантиссы и экспоненты от 10**-38 до 10**38.

Кроме того Turbo Pascal поддерживает стандарт IEEE 754 для двоичной арифметики с плавающей точкой. В этом случае добавляются типы данных с Single, Double, Extended и Comp:

Single - размер 4 байта, допускается 7 значащих цифр и диапазон экспоненты от 10**-45 до 10**38;

Double - размер 8 байт, допускается 15 значащих цифр и диапазон экспоненты от 10**-324 до 10**308;

Extended - размер 10 байт, допускается 19 значащих цифр и диапазон экспоненты от 10**-4951 до 10**4931.

Таблица 2.2 Вещественные типы данных.

Тип	Диапазон значений	Размер в байтах
Real	12.9x10E-39..1.7x10E38	6 б (вещественный)
Single	1.5x10E-45..3.4x10E38	4 б (с одинарной точностью)
Double	5.0x10E-324..1.7x10E308	8 б (с двойной точностью)
Extended	1.9x10E-4951..1.1x10E4932	10 б (повышенной точностью)
Comp	-2E+63+1..2E+63-1	8 б (сложный)

*Comp - только целые значения

Вызовем редактор Turbo Pascal и введем программу, иллюстрирующую применение некоторых типов данных:

```
Program DoRatio;
var
  A, B: Integer;
  Ratio: Real;
Begin
   Write('Enter two numbers: ');
   Readln(A, B);
   Ratio := A div B;
   Writeln ('The ratio is ', Ratio)
end.
```

Сохраним ее в файле DORATIO.PAS с помощью функции основного меню File/Save As. Нажмите ALT-R для компиляции и запуска программы. Введем два значения, 10 и 3, и получим результат 3.000000.

Ожидая ответа 3.3333333333, мы получили результат 3. Получили такой результат, потому что был использован оператор "div" для деления целых чисел (остаток отбрасывается).

Исправим оператор "div" на

```
Ratio := A / B;
```

сохраним программу (F2), откомпилируем и выполним. Новый результат 3.3333333333, как и ожидали. Оператор деления "/" обеспечивает более точный результат.

Символьные и строковые типы данных.

Паскаль позволяет определять тип Char, имеющий размер в один байт и содержащий один символ. Символьные константы содержат один символ, заключенный в апострофы ('A','e','r','2'). Заметим, что '2' означает символ 2, а 2 означает целое число 2 (и 2.0 - вещественное число) .

```
Program DoRatio;
Var
  A, B: Integer;
  Ratio: Real;
  Ans: Char;
Begin
 Repeat
  Write ('Введите два числа: ');
  Readln (A, B);
  Ratio := A / B;
  Writeln ('Коэффициент равен ', Ratio);
  Write ('Повторить? (Y/N) ');
  Readln (Ans);
  Until UpCase (Ans) = 'N';
End.
```

После вычисления коэффициента, выдается сообщение:

(Повторить? (Y/N).

Программа находится в состоянии ожидания одного из указанных символов и нажатия клавиши "Enter". Если Вы нажмете "n" или заглавную "N", то условие "until" будет выполнено и цикл закончится. В случае, если Вы ответите "Y" ("y"), то выполнение цикла будет продолжено.

"N" и "n" не одно и тоже, потому что они имеют различные значения в коде ASCII. Каждый символ имеет свой код ASCII, представленный 8-битным значением (символ занимает 1 байт).

В Turbo Pascal существуют два дополнительных способа описания символьных констант: с помощью символа "^" и символа числа "#". Символы с кодами от 0 до 31 - управляющие символы. Они обозначаются аббревиатурами (CR - возврат каретки, LF - перевод строки, ESC - выход) или с помощью двух клавиш, одна из которых Ctrl + буква. Значение буквы получается прибавлением 64 к управляющему коду.

Например, управляющий символ кода ASCII 7 известен как Bel или Ctrl+G. Turbo Pascal дает возможность представить символы с помощью "^", за которым следует буква (или символ). Так, "^G" то же самое, что и Ctrl-G. "^G" можно использовать в операторах Turbo Pascal: Writeln (^G). Этот метод применим только к управляющим символам.

Вторая возможность обозначения - использование символа номера "#", за которым следует код ASCII. Так, #7 то же самое, что и "^G", #65 - 'A', а #233 - один из специальных псевдографических символов IBM PC.

В большинстве случаев используются строки символов. Рассмотрим программу

```
program Hello;
var
  Name: String(30);
 begin
```

```
    Write ('Ваше имя? ');
    Readln (Name);
    Writeln ('Привет, ', Name)
end.
```

Переменная Name объявлена, как string (строка), и под нее резервируется 30 байт (для 30 символов). Кроме того, Turbo Pascal отводит еще один байт, в котором содержится текущая длина строки. В этом случае, независимо от того, какой длины будет введено имя, оператор Writeln распечатает имя указанной длины. Если будет введено имя больше 30 символов, то будут использоваться только первые 30 символов, а остальные будут проигнорированы.

При описании строковой переменной можно указывать ее размер, но не более 255. По умолчанию длина строковой переменной - 255 символов.

Turbo Pascal предлагает несколько процедур и функций, работающих со строковыми переменными. Их описание дано в главе 1 Справочного руководства по библиотеке.

Булевские данные.

Встроенный тип Boolean данных в Turbo Pascal имеет два возможных значения: True и False (истина и ложь). Можно объявить переменную типа Boolean, и присвоить ей значение True или False, или же, что более важно, присвоить ей значение выражения, которое при вычислении принимает одно из этих значений.

Булевское выражение - это выражение, которое принимает значение True или False. Оно состоит из логических выражений, использующих операции отношений (<, >, < >, <=, >=, =), логических функций (AND, OR, NOT), булевских переменных и/или других булевских выражений. Например:

```
while (Index <= Limit) and not Done do...
```

while содержит булевское выражение. Булевское выражение в этом случае все, что находится между ключевыми словами while и do. Done - это переменная (или функция) булевского типа.

Идентификаторы.

До сих пор в примерах содержались идентификаторы переменных без описания правил по которым они образуются. Рассмотрим их более подробно.

Имена, которые даются константам, типам данных, переменным, функциям, называются идентификаторами.

Идентификаторы в наших примерах использовались для:

1. Обозначения встроенных типов данных Integer, Real, String ;
2. Названия программ Hello, DoSum, Ratio ;
3. Переменных, определенных в программах Name, A, B, Sum, Ratio ;
4. Имен встроенных процедур Write, Writeln, Readln.

Turbo Pascal имеет несколько правил образования идентификаторов. Краткое описание правил:

- Все идентификаторы начинаются с буквы или знака подчеркивания '_' (a..z, A..Z, _). Последующими символами могут быть буква, знак подчеркивания, цифра (0-9). Другие

символы недопустимы.

- Идентификаторы рассматриваются без учета регистров (прописные и строчные буквы не различаются). Это значит, что a..z тождественно A..Z. Например? index тоже самое что и Index, и INDEX.

- Идентификаторы могут иметь различную длину, но используются только первые 63 символа.

Операторы.

Итак, данные получены программой и присвоены переменным. Программе нужно их обработать и получить результат, используя операторы.

Существует следующие типы операторов:

- присваивания,
- арифметические,
- побитовые,
- отношений,
- логические,
- над множествами,
- над строками.

Большинство операторов в Паскале бинарные, имеют два операнда; унарные операторы имеют один операнд. Бинарные операторы имеют обычно алгебраическую форму, например a+b. Унарный оператор предшествует своему операнду, например -b.

В сложных выражениях порядок выполнения операций определяется правилами приоритета (см. таблицу 2.3)

Таблица 2.3 Приоритет операторов.

Операторы	Приоритет	Категория
@, not	Первый (высший)	Унарные
*, /, div, mod, and, shl, shr	Второй	Мультипликативные
+, -, or, xor	Третий	Аддитивные
=, <>, <, >, <=, >=, in	Четвертый (низший)	Отношения

Операции равного приоритета выполняются слева направо, хотя компилятор может их перегруппировать для генерации оптимального кода.

Последовательность операторов с одинаковым приоритетом выполняется слева направо. Выражения, заключенные в скобки, вычисляются в первую очередь, независимо от предшествующих и последующих операторов.

Операторы присваивания.

Основной операцией является операция присваивания:

Ratio := A/ B.

В Паскале операция присваивания - это комбинация двоеточия и знака равенства: ":=". В примере значение выражения A /B, стоящее справа от ":=", присваивается переменной Ratio, стоящей слева.

Порядок выполнения этого оператора следующий:

1. Сначала вычисляется значение выражения стоящего справа от знака присваивания (если оно существует). Для этого из памяти ЭВМ извлекаются значения переменных, входящих в состав выражения и выполняются все необходимые действия.

2. Затем, полученное значение присваивается переменной, имя которой записано слева от знака присваивания.

Поэтому в программировании в отличие от математики имеют смысл выражения следующего типа:

A := A+1 или S := S + X

Арифметические операторы.

Паскаль поддерживает обычный стандартный набор бинарных арифметических операторов, которые выполняются над целыми и вещественными числами:

- умножение (*);
- целочисленное деление (**div**);
- деление вещественных с остатком (/);
- остаток от деления (**mod**);
- сложение (+);
- вычитание (-).

Кроме того, поддерживаются унарные операторы:

- унарный минус (a+(-b)) - дополнение до двух;
- унарный плюс (a+(+b)) - ничего не выполняет, введен для полноты.

Побитовые операторы.

Для операций над битами в Паскале имеются следующие операторы:

- **shl** (shift left). Сдвигает биты влево на указанное число бит, заполняя оставшиеся справа разряды нулями
- **shr** (shift right). Сдвигает биты вправо на указанное число бит, заполняя оставшиеся слева разряды нулями
- **and**. Выполняет логическое and (и) над парой битов, возвращает 1 , если оба бита 1 и 0 в противном случае
- **or.** Выполняет логическое or (или) над парой битов, возвращает 0 , если оба бита равны 0 и 1 в противном случае
- **xor.** Выполняет логическое исключающее или над парой битов, возвращает 1, если биты имеют разное значение и 0 в противном случае
- **not.** Операция логического дополнения бита заменяет 0 на 1 и обратно.

Эти операторы выполняют действия на низком уровне с целочисленными значениями .

Операторы отношений.

Операторы отношений сравнивают два значения, возвращая в результате булевское значение True или False. В Паскале реализуются операторы:

> - больше;
>= - больше или равно;
< - меньше;
<= - меньше или равно;
= - равно;
< > - не равно;
in - является элементом множества.

Эти операторы входят в состав логических выражений, результатом вычисления которых могут быть лишь два значения: **True** или **False**.

Введем в качестве примера следующую программу:

```
program TestGreater;
var
  A, B: Integer;
  Test: Boolean;
Begin
  Write ('Введите два числа: ');
  Readln (A, B);
  Test := A > B;
  Writeln ('A больше чем B ', Test);
end.
```

Результат программы : True, если вы введете с клавиатуры значение A больше, чем B , и False , если A меньше или равно B.

Логические операторы.

В Паскале есть четыре логических оператора - **and, xor, or, not**. Они аналогичны побитовым операторам, но имеют свои отличия.

Эти логические операторы работают с логическими значениями (True -«логическая 1» и False – «лог. 0»), позволяя комбинировать выражения отношений, булевские переменные и булевские выражения.

Различия между этими операторами и побитовыми операторами следующие:

- Логические операторы возвращают результат True или False (булевское значение), в то время как побитовые операторы производят действие над целыми значениями.

- Эти операторы не позволяют комбинировать булевские и целые выражения; другими словами, выражение Flag and Indx недопустимо, если Flag - булевский тип, а Indx - целый тип (или наоборот).

- Логические операторы and и or имеют короткую форму вычисления по умолчанию, а xor и not - нет. Допустим, имеется выражение exp1 and exp2. Если exp1 - False, то все выражение имеет значение False и выражение exp2 не вычисляется. Аналогично, в выражении exp1 or exp2, exp2 не будет вычисляться, если exp1 - True. Можно установить полную схему вычисления булевских выражений, используя опции Complete Boolean Eval (Options/ Compiler).

Адресные операторы.

В Паскале поддерживаются два специальных оператора над адресами: вычисление адреса (@) и оператор косвенной ссылки (^).

Оператор @ возвращает адрес заданной переменной; если Sum переменная целого типа, то @Sum - адрес в памяти этой переменной.

Операторы над множествами.

Операторы над множествами выполняются в соответствии с правилами логики теории множеств. Они включают:

- + - объединение;
- - - разность;
- * - пересечение.

Строковые операторы.

Существует единственный оператор - "+", который выполняет конкатенацию (слияние) двух строк.

Вывод информации из памяти ЭВМ.

Может показаться странным , что речь о выводе пойдет прежде, чем о вводе, но программа, которая не выводит какую-либо информацию, не имеет смысла. Вывод обычно принимает форму, которая зависит от выходного устройства:

- на экран (слова и изображения),
- на запоминающие устройства (дискеты и винчестер),
- в порты ввода/вывода.

Процедура Writeln.

Вы уже использовали наиболее распространенную функцию Паскаля - подпрограмму **Writeln**. Назначение ее - запись (вывод) информации на экран. Ее формат прост и гибок при использовании:

Writeln (элемент, элемент,....);

Каждый элемент в списке вывода - это то, что Вы хотите вывести на экран и может быть:

- значением , таким как целое или вещественное число (3, 42, - 1732.3),
- символом ('a','Z'),
- строкой ('Здравствуй, мир'),
- булевским значением (True).

Кроме того, в списке вывода может быть

- именованная константа (имя константы),
- переменная,
- вызов функции, если она возвращает значение целого типа, вещественное число, символ, строку или булевский тип.

Все элементы печатаются в строку в заданном порядке. После вывода курсор устанавливается на начало следующей строки. Если есть необходимость оставить курсор в этой

же строке после последнего элемента, то используйте

Write (элемент, элемент,...);

При выводе элементов списка вывода процедуры Writeln между ними автоматически пробелы не вставляются. При желании можно установить их самим:

Writeln (элемент,'_',элемент,'_',....);

Примеры применения оператора вывода:

	на экране
A:=1; B:=2; C:=3;	появится:
Name := 'Frank';	
Writeln (A, B, C);	123
Writeln (A, ' ', B, ' ', C);	1 2 3
Writeln ('Hi' , Name);	HiFrank;
Writeln('Hi, ',' , Name, '.');	Hi,Frank.

Так же можно использовать параметры определения ширины поля на экране монитора, которое будет отводиться для данного элемента. В этом случае оператор имеет формат:

Writeln (элемент: длина,....);

где длина - целое выражение (константа, переменная, вызов функции), определяющее общий размер поля для вывода элемента.

Рассмотрим следующую программу и полученный в результате вывод:

A:=10; B:=2; C:=100;	
Writeln(A, B, C);	102100
Writeln(A:2, B:2, C:2);	10 2100
Writeln(A:3, B:3, C:3);	10 2100
Writeln(A, B:2, C:4);	10 2 100

Заметим, что элемент дополняется начальными пробелами слева в соответствии с указанной длиной. Само значение выравнивается по правому краю отведенного поля.

Что, если размер поля меньше, чем необходимо? Во втором операторе Writeln вышеприведенного примера для C=100, длина поля меньше, чем нужно, т.е. задано 2, нужно 3. При выводе Паскаль увеличивает размер до минимально необходимого.

Этот метод применим для всех допустимых элементов: целого типа, вещественных чисел, символов, строк и булевских типов. Однако, при указании ширины (размера) поля для вещественных чисел выравнивание происходит слева и распечатывается в экспоненциальной форме.

x:=421.53;	
Writeln (x);	4.2153000000E+02
Writeln (x:8);	4.2E+02

Поэтому, Паскаль позволяет добавить второй операнд длины:

элемент : длина : количество цифр.

Второе число указывает, сколько цифр выводить для числа с фиксированной точкой после точки:

```
x:=421.53;
Writeln (x:6:2);        421.53
Writeln (x:8:2);        421.53
Writeln (x:8:4);        421.5300
```

Ввод информации в память ЭВМ

В стандартном Паскале есть две основных функции ввода информации **Read** и **Readln**, которые используются для чтения данных с клавиатуры.

Их формат:

Read (элемент, элемент,...);

Readln (элемент, элемент,...);,

где каждый элемент - это переменная целого, вещественного, символьного типа или строка. Числа при вводе с клавиатуры должны отделяться друг от друга пробелами или нажатием клавиши Enter.

Порядок выполнения оператора ввода:

1. Когда в программе встречается оператор ввода выполнение программы приостанавливается и на экране появляется мигающий курсор в том месте, где был произведен последний вывод информации на экран.

2. Затем, с помощью клавиатуры пользователь набирает данные, нажимая пробел или Enter после каждого элемента.

3. После ввода значения последнего элемента данные заносятся в память ЭВМ и работа программы продолжается со следующего оператора.

Оператор ветвления

Иногда бывает необходимо выполнить часть программы, если заданное условие имеет значение True или False, или когда заданное выражение принимает определенное значение. Посмотрим, как это реализуется в Паскале.

Оператор **if.**

Посмотрим, как оператор if использовался в предыдущих программах, отметив, что его общий формат:

if *выражение* **then** *оператор 1* **else** *оператор 2*,

где *выражение* - любое булевское выражение (вырабатывающее в результате True или False); *оператор 1* и *оператор 2* - операторы Паскаля. Если выражение принимает значение True, то выполняется оператор 1; в противном случае - оператор 2.

Два важных момента, на которые следует обратить внимание при использовании if/then/else.

Во-первых, ветвь **else** не является обязательной, другими словами, допустимо использовать оператор **if** в следующем виде:

if *выражение* **then** *оператор 1*

В этом случае *оператор 1* выполняется только тогда, когда выражение имеет значение True. В противном случае пропускается *оператор 1* и выполняется следующий оператор.

Во-вторых, если необходимо выполнить более одного оператора, в случае, когда выражение принимает значение, True или False, то следует использовать составной оператор.

Составной оператор - это ключевое слово **begin**, несколько операторов разделенных точкой с запятой и ключевое слово **end**.

В примере используется один оператор в основной ветви и составной оператор в предложении **else**:

```
if B = 0.0 then
  Writeln ('деление на нуль невозможно.');
else
  begin
    Ratio := A / B;
    Writeln('Отношение =' , Ratio)
  end;
```

Оператор выбора **case**.

Оператор **case** - мощное средство выбора альтернатив. Позволяет уменьшить количество операторов if.

Оператор **case** состоит из выражения (ключа выбора) и списков операторов, каждому из которых предшествует метка того же типа, что и ключ выбора. Это значит, что в данный момент выполняется тот оператор, у которого значение совпадает с текущем значением ключа выбора. Если совпадения значений не происходит, то не выполняется ни один из операторов, входящих в **case** или же выполняются операторы, стоящие после необязательного слова **else** (else - расширение стандартного Паскаля).

Метка case состоит из любого количества констант или поддиапазонов, разделенных запятыми, за которым следует двоеточие (:), например:

```
case BirdSight of
  'C', 'c': Curlens := Curlens + 1;
  'H', 'h': Herons := Herons + 1;
  'E', 'e': Egrets := Egrets + 1;
  'T', 't': Terns := Terns + 1;
end; {case}
```

Диапазон записывается в виде двух констант, разделенных двумя точками "..". Тип константы должен соответствовать типу ключа выбора. Оператор, стоящий после двоеточия (:), выполняется в том случае, если значение ключа выбора совпадает со значением константы или, если его значение попадает в диапазон значений.

Операторы цикла

В случае, когда при истинности какого-либо условия, необходимо выполнять группу операторов повторно, используются циклы.

Существует три основных вида циклов: цикл while, цикл repeat и цикл for. Рассмотрим их.

Цикл **while** («до»)

Цикл while используется для проверки некоторого условия в начале цикла. Введите следующую программу:

```
program Hello;
var
  Count: Integer;
begin
  Count := 1;
  while(Count <= 10) do
       begin
         Writeln ('Здравствуй и прощай!');
         Inc(Count);
       end;
  Writeln ('Это конец');
end.
```

Комментарии к данной программе

1. Сначала переменной Count присвоится значение равное 1.
2. Затем, при входе в цикл проверяется условие: значение Count меньше или равно 10.
3. Если да, то выполняется **тело цикла** (операторы, находящиеся между ключевыми словами begin....end.)
4. На экран выводится сообщение "Здравствуй и прощай".
5. Значение Count увеличивается на 1.
6. Возврат на начало цикла. Значение Count проверяется заново и тело цикла выполняется вновь, до тех пор пока значение переменной Count удовлетворяет условию.
7. Как только значение Count становится равным 11, цикл завершается, и на экран выводится сообщение "Это конец".

Формат оператора цикла while:

while *выражение* **do** *тело цикла*;

В цикле while вычисляется выражение. Если оно имеет результат - True, выполняется тело цикла. В противном случае выполнение цикла завершается.

Цикл **Repeat....Until** («пока»)

Второй цикл repeat...until рассмотрим на примере программы DORATIO.PAS:

```
program DoRatio;
var
  A, B: Integer;
  Ratio: Real;
  Ans: Char;
begin
   repeat
   Write('Введите два числа');
   Readln(A, B);
   Ratio := A / B;
   Writeln('Отношение равно', Ratio);
   Writeln('Повторить? (Y/N)');
```

```
  Readln(Ans);
  until Upcase (Ans) = 'N';
end.
```

Как описывалось ранее, в этой программе повторяется выполнение операторов, пока ответ на вопрос - n или N (Повторить? Y/N). Другими словами repeat и until, повторяются, до тех пор, пока значение выражения при until не будет True.

Формат цикла:

repeat
тело цикла;
until *выражение*.

Существуют два основных отличия от цикла while:

1. операторы в цикле repeat выполняются хотя бы один раз, потому что проверка выражения осуществляется в конце тела цикла. В цикле while, если значение выражения False, тело его пропускается сразу.

- цикл repeat выполняется пока выражение не станет True, в то время, как цикл while выполняется до тех пор, пока выражение имеет значение True. При замене одного типа цикла на другой необходимо на это обращать особое внимание.

Рассмотрим программу HELLO, где цикл while заменен на цикл repeat:

```
program Hello;
var
  Count: Integer;
begin
  Count := 1;
  repeat
   Writeln ('Здравствуй и прощай!');
   Inc (Count);
  until Count > 10;
Writeln ('Это конец');
end.
```

Отметим, что теперь переменная Count проверяется на значение больше 10 (а в while было Count <= 10).

В заключение, в цикле repeat может использоваться просто группа операторов, а не составной оператор. При использовании этого цикла не записываются слова **begin...end**, как в случае с циклом while.

Запомните, что цикл repeat выполнится хотя бы один раз, в то время, как цикл while может ни разу не выполниться в зависимости от значения выражения.

Цикл **for** («для»).

Цикл for существует во многих языках программирования. В Паскале тоже. Однако, вариант этого цикла в Паскале как эффективен, так и ограничен.

Обычно, набор операторов выполняется фиксированное число раз, пока переменная цикла (индексная) принимает значение в указанном диапазоне. Модифицируем знакомую программу Hello следующим образом.

```
program Hello
var
```

```
  Count: Integer;
begin
  for Count := 1 to 10 do Writeln('Здравствуй и прощай!');
  Writeln('Это конец');
end.
```

При выполнении этой программы видно, что цикл for выполняется так же, как и циклы while и repeat. Фактически эквивалентно циклу while.

Формат цикла for:

for *индекс*:=выражение1 **to** выражение2 **do** *тело цикла*

где *индекс* - скалярная переменная (целого типа, символьного, булевского и любого перечислимого типа); *выражение1 и выражение2* - выражения типа, совместимого с типом индекса; *тело цикла* - одиночный или составной оператор. Индекс увеличивается на 1 после каждого выполнения цикла. Индекс можно уменьшать на 1. Для этого ключевое слово **to** заменяется на **downto**.

Цикл for эквивалентен следующей программе с циклом while:

```
  index :=exprl;
  while index <= expr2 do
   begin
     {тело цикла};
     Inc(index)
   end;
```

Главный недостаток цикла for - это возможность уменьшить или увеличить индекс только на 1. Основные преимущества - краткость, возможность использования символьного и перечислимого типа в диапазоне значений.

Процедуры и функции.

Вы изучили ветвления и циклическое повторение операторов в программе. Теперь посмотрим, как можно выполнить один и тот же набор команд в разных местах программы и с разными исходными данными. Оказывается можно объединить эту группу операторов в подпрограмму, которую можно вызвать по необходимости.

В Паскале есть два вида подпрограмм: процедуры и функции. Главное различие между ними - это то, что функция возвращает значение и может быть использована в выражении:

```
X := sin(A);
```

в то время, как процедура может быть вызвана :

```
Writeln ('Это проверка');
```

Однако перед знакомством с процедурами и функциями, необходимо рассмотреть структуру программ.

Структура программ.

В стандартном Паскале программы имеют жесткий формат:

program *имя программы;*
label метки;
const *объявление констант*;

type *определение типов данных;*
var *объявление переменных*;
procedure и function;
begin
тело программы
end.

Наличие всех пяти секций объявлений - *label, const, type, var, procedure и function* - в Вашей программе необязательно. Однако для стандартного Паскаля, если они присутствуют, порядок их следования строго регламентирован, и в программе они должны присутствовать только один раз. За секцией объявлений, следуют процедуры и функции, и только затем тело программы.

Turbo Pascal обеспечивает более гибкую структуру программы. Главное - это чтобы ключевое слово *program* должно быть первым, а *тело программы* последним. Порядок описания остальных секций жестко не регламентирован, но идентификаторы должны быть объявлены до их использования во избежание ошибок компиляции.

Структура процедуры и функции.

Процедуры и функции, известные под общим именем как подпрограммы могут быть описаны в любом месте программы, но до тела главной программы. Формат процедур:

procedure *имя процедуры (параметры);*
label *метки*;
const *объявление констант*;
type *определения типов данных*;
var *объявления переменных*;
procedure и function;
begin
тело главной процедуры;
end;

Функции имеют такой же формат, как и процедуры, только они начинаются с заголовка function и заканчиваются типом данных возвращаемого значения:

function *имя функции* (*параметры*): *тип данных*;

Имеются только два отличия процедур и функций от программ:
- процедуры и функции имеют заголовок procedure или function, соответственно, а не program;
- процедуры и функции заканчиваются точкой с запятой (;), а не точкой (.). Процедуры и функции могут иметь описания своих констант, типов данных, переменных и свои процедуры и функции. Но все эти элементы могут быть использованы только в тех процедурах и функциях, где они объявлены (локальные переменные).

Пример программы с пользовательской процедурой.
Рассмотрим версию программы DORATIO, в которой используются процедура получения двух значений и функция, определяющая их отношение:

```
program DoRatio;
```

```
var
  A, B: Integer;
  Ratio: Real;
procedure GetData (var X, Y: Integer);
begin
  Writeln ('Введите два числа:');
  Readln (X, Y);
end;
function GetRatio (I, J: Real);
begin
  GetRatio (I / J);
end;

begin
  GetData (A, B);
  Ratio := GetRatio (A, B);
  Writeln ('Отношение равно ', Ratio);
end.
```

Это, конечно, не улучшение первоначальной программы, так как она имеет больший размер и медленнее выполняется. Но она показывает как используются и работают процедуры и функции.

После компиляции и запуска программы первым выполняется оператор GetData(A,B). Этот тип оператора известен как вызов процедуры. При обработке вызова выполняются операторы в GetData, при этом X и Y (формальные параметры) заменяются на A и B (фактические параметры), другими словами при вызове процедуры происходит передача данных из основной программы в процедуру. Ключевое слово var перед X и Y в операторе вызова GetData говорит о том, что фактические параметры должны быть переменными и что значения переменных могут быть изменены и возвращены вызывающей программе. При завершении работы GetData управление возвращается в главную программу на оператор, следующий за вызовом GetData.

Следующий оператор - вызов функции GetRatio. Отметим некоторые отличия.

Во-первых, GetRatio возвращает значение, которое должно быть использовано; в этом случае, оно присваивается Ratio.

Во-вторых, значение присваивается GetRatio в главной программе; этим функция определяет, какое значение возвращается.

В-третьих, нет ключевого слова var перед формальными параметрами I и J. Это означает, что они могут быть любыми целочисленными выражениями, такими как Ratio:= GetRatio(A+B,300); и что если даже их значения будут изменены в функции, то новые значения не возвратятся обратно в вызывающую программу. Кстати, это не является отличием процедуры от функции. Можно использовать оба типа параметров для обоих типов программ.

Комментарии.

Иногда бывает необходимо вставить в программу замечания, напоминающие или информирующие о том, что означает переменная, какие действия выполняет функция или оператор. Эти замечания называют комментариями. Паскаль позволяет вставлять в программу сколько угодно комментариев.

Комментарий начинается левой фигурной скобкой «{». Она указывает компилятору: игнорировать все, пока не встретится правая фигурная скобка «}».

Комментарий может занимать несколько строк:

```
{ Это пример длинного комментария,
  занимающего несколько строк}
```

Кроме того, существует альтернативная форма комментария. Начинается "(*" и заканчивается "*)". Комментарий, начинающийся с "(*" игнорирует все фигурные скобки, и наоборот.

Глава 3.

МОДУЛИ TURBO PASCAL.

Для того, чтобы создавать сложные программы, необходимо иметь понятия о модулях, которые содержат необходимые операторы, процедуры и функции, аппаратуре компьютера в достаточной степени, чтобы работать с ними. В этой главе описано, что такое модуль, как его использовать, какие модули доступны.

Что такое модули?

В Turbo Pascal возможен доступ к большому числу встроенных констант, типов данных, переменных, процедур и функций. Количество различных процедур велико, но почти никогда они все сразу в программах не используются. Все эти программы разделены на связанные между собой группы, называемые модулями, и Вы можете использовать только те модули, которые Вам необходимы.

Модуль - это набор констант, типов данных, переменных, процедур и функций. Каждый модуль аналогичен отдельной программе; т.е имеет: главное тело, которое вызывается перед стартом Вашей программы и производит необходимые действия по инициализации аппаратных средств компьютера, когда это требуется. Короче говоря, каждый модуль - это библиотека констант, процедур, функций, которую можно вставить и использовать внутри своей программы, причем программа и модули компилируются отдельно. Такая возможность дает четыре серьезных преимущества:

1. Можно поручить написание модулей для новой программы разным людям, сообщив каждому работнику, какие действия должны выполнять процедуры, входящие внутрь модуля, какие необходимы исходные данные и выходные данные. Время написания программы существенно сокращается.

2. Поскольку процедуры и функции из модулей могут вызываться многократно, то уменьшится и размер программы.

3. Повышается наглядность программы, т.к. она разделена на практически независимые части, и человеку легче понять как работают эти отдельные части, чем вся программа целиком.

4. Процедуры и функции, входящие в состав модулей могут быть использованы вами при случае и в других программах, т. е. Не надо будет разрабатывать их заново, если они универсальны.

Процедуры и функции внутри модуля связаны друг с другом. Например, модуль **Crt** содержит все необходимое для программ работающих с экраном PC.

Turbo Pascal предоставляет шесть стандартных модулей **System, Overlay, Graph, DOS, Crt, и Printer,** которые осуществляют поддержку Ваших программ на Turbo Pascal; все они сохранены в файле **TURBO.TPL**. Кроме этого существует модуль управления графическим экраном **Graph**, который хранится отдельно в файле **GRAPH.TPU.**

Структура модуля и порядок его создания – эти темы выходят за рамки школьного курса и не описаны в данной книге.

Как используются модули?

Модули, которые использует Ваша программа, уже откомпилированы и хранятся в специальном машинном коде. Более того, стандартные модули хранятся в специальном файле TURBO.TPL и автоматически загружаются в память при запуске Turbo Pascal.

В результате подключения модулей к программе увеличивается время и компиляции

программы (незначительно, приблизительно на 1 секунду). Если модули загружаются из отдельных дисковых файлов может потребоваться дополнительное время из-за чтения с диска.

Для использования модулей необходимо, чтобы в начале присутствовало ключевое слово **uses**, за которым следует список имен всех модулей, разделенных запятыми.

```
program MyProg;
  uses crt, printer;
```

При компиляции этой информации к вашей программе добавится машинный код указанного модуля. Порядок описания модулей в предложении uses не имеет большого значения.

Если предложение uses отсутствует, Turbo Pascal подсоединяет стандартный модуль **System**. Этот модуль обеспечивает выполнение некоторых стандартных подпрограмм Turbo Pascal и программ, специфичных для Turbo Pascal.

Если Вы включили модуль в свою программу, то все константы, типы данных, переменные, процедуры и функции, объявленные в интерфейсе этого модуля становятся доступными для Вашей программы.

Стандартные модули.

Файл TURBO.TPL содержит все стандартные модули, кроме Graph: System, Overlay, Crt, Dos и Printer. Эти модули загружаются в память вместе с Turbo Pascal; они всегда доступны для любой программы. Файл TURBO.TPL хранится в том же каталоге, что и TURBO.EXE (или TPC.EXE).

System.

Модуль System содержит все стандартные и встроенные процедуры и функции Turbo Pascal. Любая подпрограмма Turbo Pascal, не являющаяся стандартной и не находящаяся ни в одном другом модуле, находится в System. Этот модуль присоединяется к каждой программе.

Dos.

Dos определяет многочисленные процедуры и функции Turbo Pascal, которые эквивалентны наиболее часто используемым вызовам Dos, таким как GetTime, SetTime, DiskSize и т.д. Кроме того, здесь определяются две программы низкого уровня - MsDos и Intr, которые позволяют использовать любой вызов MS-DOS или системные прерывания. Registers - тип данных для параметров в MsDos и Intr. Кроме того, определяются некоторые другие константы и типы данных.

Overlay.

Модуль Overlay обеспечивает поддержку системы оверлеев (подгружаемых в процессе работы программ, для которых не хватает памяти в обычном режиме).

Crt.

Crt обеспечивает набор специальных средств для ввода/вывода на PC: констант, переменных и программ. Их можно использовать для работы с экраном (работа с окнами, управление курсором, управление цветом). Есть возможность вводить с клавиатуры и управлять звуковым сигналом.

Printer.

В модуле Printer объявляется переменная текстового файла **LST**, которая связывается с драйвером устройства, позволяя посылать стандартный вывод на печатающее устройство, используя Write и Writeln. Например, включив модуль Printer в программу, можно сделать следующее:

```
write (Lst,'Сумма ',A:4,' и ',B:4,' равна ');
C:=A+B;
writeln (Lst,C:8);
```

На бумаге появится запись: *Сумма A и B равна*, а далее – результат.

Graph.

Этот модуль не входит в файл **TURBO.TPL**, но должен находиться в том же каталоге, где и вспомогательные файлы, расширения которых .BGI и .CHR. Поместите **GRAPH.TPU** в текущий каталог или используйте каталог модулей для указания полного пути до GRAPH.TPU. (Если Вы используете жесткий диск и программу Install, Ваша система уже установлена так, что Вы можете использовать Graph). Модуль Graph - это набор быстродействующих эффективных графических подпрограмм, которые позволяют в полной мере использовать графические возможности PC. Этот модуль реализует независимый от устройства графический драйвер, поддерживающий графические адаптеры CGA, EGA, Hercules, AT&T400, MCGA, 3270 PC, VGA и 8514.

Глава 4.

Отладка программ пользователя в TURBO PASCAL.

В помощь пользователю Turbo Pascal предоставляет средства, необходимые для отладки его программы, способствующие устранению всех ошибок в программе, ее тщательному тестированию и выполнению. Turbo Pascal позволяет легко определять местоположение ошибок во время компиляции и во время выполнения программы.

Особенно важно то, что Turbo Pascal имеет мощный и гибкий отладчик, который позволяет пользователю выполнять программу построчно, просматривать выражения и модифицировать переменные по мере необходимости.

Типы ошибок.

Существует три основных типа программных ошибок: ошибки времени компиляции (синтаксические), ошибки времени выполнения и логические ошибки.

Ошибки компиляции.

Ошибки компиляции или синтаксические ошибки встречаются, когда забывают объявить переменную, передают ошибочное количество параметров процедуры, при присваивании вещественного значения целочисленной переменной. Это означает, что записываются операторы, которые не согласуются с правилами Паскаля.

Turbo Pascal не закончит процесс компиляции программы пользователя (генерацию машинного кода), пока все синтаксические ошибки не будут удалены. Если Turbo Pascal обнаружит синтаксическую ошибку во время компиляции программы, он останавливает компиляцию, входит в исходный текст, указывает местоположение ошибки курсором и выводит сообщение об ошибке в окно Edit. Как только пользователь исправит ошибку, он сможет начать процесс компиляции снова.

Ошибки времени выполнения.

Другой тип ошибок - ошибки времени выполнения программы или семантические ошибки. Они встречаются, когда пользователь компилирует синтаксически корректную программу, которая пытается сделать что-нибудь запрещенное во время ее выполнения, например, открывает несуществующий файл для ввода или производит деление на 0. В этом случае Turbo Pascal выводит на экран следующее сообщение об ошибке: Runtime error ## at seg:ofs (Ошибка выполнения # в сегменте: смещение) и останавливает выполнение программы пользователя.

При использовании интегрированной среды Turbo Pascal определяет местоположение ошибки выполнения автоматически, осуществляя переход в окно редактирования соответствующего исходного файла.

Логические ошибки.

Программа пользователя может содержать и логические ошибки. Это означает, что программа делает то, что ей указали вместо того, что хотелось бы. Может отсутствовать инициализация переменной (т. е. у переменной может не быть начального значения); могут оказаться ошибочными вычисления; рисунки, изображенные на экране, выглядят неправильно; программа может просто работать не так, как было задумано.

Такие ошибки находятся с большим трудом, и интегрированный отладчик поможет вам в этом случае наилучшим образом.

Интегрированный отладчик Turbo Pascal.

Некоторые ошибки времени выполнения (логические ошибки) незаметны и трудны для прослеживания. Другие ошибки могут скрываться за неуловимым взаимодействием разделов большой программы. В этих случаях необходимо интерактивное выполнение программы, во время которого производится наблюдение за значениями определенных переменных или выражений. Вам хотелось бы, чтобы Ваша программа останавливалась при достижении определенного места так, чтобы просмотреть, как она проработала этот кусок. Вам хотелось бы остановиться и изменить значения некоторых переменных во время выполнения программы, изменить определенный режим или проследить за реакцией программы. И вам хотелось бы сделать это в режиме, когда возможно быстрое редактирование, перекомпилирование и повторное выполнение программы.

Обзор возможностей отладчика:

1. Трассировка. F7 (Run/Trace Into)

Вы можете выполнить одну строку вашей программы, затем прерваться и посмотреть на результаты на экране (Alt+F5). При вызове процедуры или функции внутри вашей программы, Вы можете задать режим выполнения вызова как одного шага или режим трассировки этой процедуры или функции строка за строкой.

2. Выполнение до курсора F4 (Run/Go to Cursor)

Вы можете передвинуть курсор на определенную строку в Вашей программе, а затем указать отладчику выполнить программу до достижения этой строки. Это позволяет обходить циклы или другие утомительные участки программы, это также позволяет перебираться в то место программы, откуда Вы хотите начать отладку.

3. Прерывание.

С помощью команды Debug/Breakpoints Вы можете пометить строки в Вашей программе как точки прерывания. Когда в процессе выполнения Вашей программы достигается точка прерывания, выполнение программы приостанавливается и отображается исходный текст и курсор останавливается на строке с точкой прерывания. Затем Вы можете проверить значения переменных, начать трассировку или выполнить программу до другой точки прерывания. Вы можете подключить условие к точке прерывания. Вы можете также прерваться в любой точке Вашей программы, нажав клавишу Ctrl-Break. Произойдет остановка на следующей строке исходной программы, как если бы в этой строке была установлена точка прерывания.

4. Наблюдение (Debug/Watches)

Пользователь имеет возможность задавать для просмотра в окне Watch некоторые объекты (переменные, структуры данных, выражения). Просматриваемые данные меняются, отражая текущие изменения в программе при пошаговом выполнении.

5. Вычисление/модификация. Ctrl-F4 (Debug/Evaluate/Modify)

Пользователь может вызвать окно Evaluate, чтобы проверить значения переменных, структуру данных и выражения в интерактивном режиме. Используя окно Evaluate, Вы можете изменить значение любой переменной, включая строки, элементы массива и поля записей. Это обеспечивает простой механизм для проверки, как Ваш код реагирует на определенную установку значений или условий.

Глава 5.

Встроенный редактор текстов программ .

Эта глава является полным справочником команд редактора Turbo Pascal. Таблица 5.1 содержит список всех команд редактора; таблицы и текст, следующие за ними, охватывают те аспекты редактора, которые нуждаются в дальнейшем пояснении.

Примечание. Контекстно-ориентированную справочную информацию всегда можно получить, нажав F1.

Кроме этого Turbo Pascal имеет поддержку мышки для многих команд передвижения курсора и пометки блока.

Меню Edit содержит команды для вырезания, копирования и вставки текста в файл, копирования примеров из окна Help в окно Edit (редактора) и просмотра содержания кармана. Если Вы сейчас запустите Turbo Pascal в первый раз, окно редактора уже будет активно. Чтобы открыть другие окна редактора, войдите в меню File и выберите Open. Из окна редактора просто нажмите F10 для получения полосы меню; чтобы вернуться в окно редактора, нажимайте Esc столько раз, пока не выйдете из меню. Если имеется мышка, нужно просто щелкнуть в любом месте окна редактора.

Когда Вы введете довольно много текста, нажимая в конце строки Enter, экран заполнится полностью и верхняя строка передвинется вверх за пределы экрана. Не беспокойтесь, она не потеряна, Вы можете вернуться назад и передвинуться далее в своем тексте с помощью клавиш управления курсором.

Редактор способен восстанавливать изменения в последней модифицированной строке (команда Edit/Restore Line)

Справочник редактора.

Редактор является гораздо более мощным средством, чем может показаться после краткого обзора. Вдобавок к выбору меню, он использует примерно 50 команд для перемещения курсора, листания страниц, нахождения и замены строк и т.д. Эти команды можно разбить на четыре основные категории:

- передвижение курсора;
- операции вставки и удаления;
- операции над блоками;
- разнообразные операции редактирования.

Таблица 5.1.
Полный обзор команд редактора.

Передвижение	Команда
Основные команды перемещения курсора	
На символ влево	Стрелка влево
На символ вправо	Стрелка вправо
На слово влево	Ctrl + Стрелка влево
На слово вправо	Ctrl + Стрелка вправо
На строку вверх	Стрелка вверх
На строку вниз	Стрелка вниз

Скроллинг вверх на одну строку	Ctrl+W
Скроллинг вниз на одну строку	Ctrl+Z
На страницу вверх	PgUp
На страницу вниз	PgDn
На большие расстояния	
Начало строки	Home
Конец строки	End
Верх окна	Ctrl+Home
Низ окна	Ctrl+End
Начало файла	Ctrl+PgUp
Конец файла	Ctrl+PgDn
Начало блока	Ctrl+Q, B
Конец блока	Ctrl+Q, K
Последняя позиция курсора	Ctrl+Q, P
Команды удаления и вставки.	
Вкл/выкл режима вставки	Insert
Удалить символ слева от курсора	Backspace
Удалить символ под курсором	Del
Удалить слово справа	Ctrl+T
Вставить строку	Ctrl+N
Удалить строку	Ctrl+Y
Удалить конец строки	Ctrl+Q, Y
Команды работы с блоками.	
Отметить блок	Shift+стрелки или Ctrl-K B, Ctrl-K K
Отметить одно слово	Ctrl+K, T
Скопировать блок в карман	Edit/Copy или Ctrl+Ins
Переместить блок в карман	Edit/Cut или Shift+Del,
Вставить блок из кармана	Edit/Paste или Shift+Ins
Удалить блок	Edit/Clear или Ctrl+Del
Прочитать блок с диска	Ctrl+K, R
Записать блок на диск	Ctrl+K, W
Спрятать/показать блок	Ctrl+K, H
Напечатать блок на принтере	File/Print или Ctrl+K, P
Сделать смещение блока вправо	Ctrl+K, I
Сместить блок влево	Ctrl+K, U
Другие команды редактора.	
Вкл/выкл автоотступа	Options/Environment/Editor/ Autoindent Mode
Найти маркер места	Ctrl+Q, n
Перейти к полосе меню	F10
Новый файл	File/New
Открыть файл	File/Open (F3)
Печатать файл	File/Print
Выход из ИСП Turbp Pascal	File/Quit (Alt+X)
Восстановить сообщение об ошибке	Ctrl-Q W
Восстановить строку	Edit/Restore Line или Ctrl+Q, L
Вернуться в редактор из меню	Esc
Сохранить исходный текст	File/Save (F2)
Найти фрагмент текста	Search/Find или Ctrl+Q, F
Найти и заменить	Search/Replace или Ctrl+Q, A

Установить маркер	Ctrl+K, n
Передвинуться с помощью табуляции	Tab
Режим табуляции	Options/Environment/Editor/ Use tab characters
Режим без отступа	Options/Environment/Editor/Backspace Unindents

---

Примечания:

1. **Слово** определено как последовательность символов, разделенных одним из следующих знаков: пробел < > . : ( ) ^ ` * + - / $ # _ = | ~ ? ! " % & @ \ { } [ ] и всеми управляющими и графическими символами.

2. Многие из команд этой таблицы можно выполнить с помощью мышки..

Перемещение курсора.

Существуют три команды передвижения курсора, которые требуют дополнительных пояснений:

Ctrl+Q, B и Ctrl+Q, K передвигают курсор к маркеру начало-блока или конец-блока. Обе эти команды работают даже в том случае, когда блок не видно.

Ctrl+Q, B работает даже в том случае, если маркер конец-блока не установлен, а Ctrl+Q, K работает даже тогда, когда не установлен маркер начало-блока.

Ctrl+Q, P передвигает курсор к последней его позиции перед последней командой. Эта команда особенно полезна, если была выполнена операция поиска или поиска-и-замены, а Вы хотели бы вернуться туда, где Вы находились перед выполнением поиска.

Блоки.

Блок - это любое количество текста, от единственного символа до сотен строк, которое должно быть помечено специальными символами маркер-блока. Одновременно в окне может быть только один блок. Блок помечается путем помещения маркера начало-блока на первый символ и маркера конец-блока на последний символ требуемой части текста. Помеченный блок можно копировать, передвигать, удалять, выводить на печать или записывать в файл.

Глава 6

Строковый тип данных STRING.

Введение

В TP тип - строка (стандартный тип string) - представляет собой последовательность символов произвольной длины (до 255). Строку можно рассматривать как массив символов, однако в связи с широким использованием строк и некоторыми особенностями по сравнению со стандартными массивами они выделены в отдельный тип данных.

Важность получения навыков работы со строками текста очевидна: большая часть информации, подвергающаяся обработке с помощью ЭВМ – текстовая. К текстовой информации можно отнести любые документы делопроизводства, базы данных, отчеты, типовые письма и многие другие документы. К наиболее часто используемым операциям обработки строк символов (или текста) относятся:

1. Поиск нужного символа или группы символов (слов, словосочетаний) в тексте.
2. Удаление из строки необходимого символа или сочетания символов.
3. Вставка внутрь строки необходимого символа или сочетания символов (слова, словосочетания).
4. Замена одного слова в строке на другое.
5. Слияние нескольких строк в одну или разбиение одной строки на две или более строк.
6. Сортировка слов или строк по алфавиту или в обратном алфавитном порядке.
7. Шифрование секретных сообщений и сжатие текстовой информации для ускорения ее передачи по сетям InterNet.

Многие из перечисленных операций уже реализованы и используются такими программными продуктами как текстовые процессоры и СУБД.

Описание строк

Определение строкового типа содержит служебное слово string после которого может в квадратных скобках указываться размер строки, например : string[56], srting[8] или просто srting (в этом случае под строку отводится размер 255 символов в строке)

Рассмотрим пример описания данных указанного типа : var str :string[80];
str1,str2,str3:string[10]; ....

Фактически строка из N символов представляет собой массив из N+1 символа:
string[N] = array[0..N] of char

Нулевой символ предназначен для указания используемого количества символов строки и может изменяться от символа с кодом 0 до символа с кодом N. С ним можно работать как и с остальными символами строки (записывать, читать его значение и т.п.), но не забывая о его основном предназначении. Таким образом каждая строка в памяти занимает столько байтов каков ее размер с учетом 0 символа, где записывается текущая длина. Каждый символ в строке занимает 1 байт памяти.

Операции ввода-вывода строкового типа :

Ввод - вывод значений переменных строкового типа осуществляется стандартно с помощью операторов read и write. Например: Ввести строку с клавиатуры.

```
var s:string[20];
begin
  writeln ('введите строку не более 20 символов');
```

```
  readln (s);   writeln (s);   writeln('Спасибо')
end.
```

На экране :

```
Введите строку не более 20 символов
фавпаврволокп влв кзк вдл
фавпаврволокп влв кз
Спасибо
```

Операции слияния строк

Операция слияния строк называется "конкатенацией" и обозначается символом "+". Пример слияния строк:

```
var
  s1:string[5];
  s2:string[10];
begin
  Writeln ('введите строку 1 ');  readln(s1);
  writeln('введите строку 2 ');   readln(s2);
  writeln(s1+s2);
end.
```

На экране:

```
введите строку 1
привет
введите строку 2
дет
приведет
```

Стандартные процедуры и функции обработки строковых величин

Функция Concat (s1,s2,...,sn) - выполняет конкатенацию последовательности строк (аналогично "+"). s1,s2,... -строки.

Функция Copy (ST,K,N) - возвращает подстроку строки st, состоящую из N символов, начиная с символа K.

Процедура Delete (ST,K,N) - удаляет из строки ST подстроку из N символов, начиная с символа K.

Процедура Insert (SUBST, ST, K) - добавляет в строку ST подстроку SUBST, начиная с символа K.

Функция Length (ST) - возвращает длину строки ST.

Функция Pos (SUBST, ST) - производит поиск подстроки SUBST в строке ST и возвращает номер позиции с которой она начинается, в противном случае - 0.

Процедура Str (X,ST) - преобразует численное значение X в его строковое представление ST (X - Real или Integer).

Процедура Val (ST,X,CODE) - преобразует строковое значение ST в его численное представление X. CODE содержит нуль, если преобразование прошло успешно или номер позиции строки ST в которой обнаружен ошибочный символ.

Функция Upcase(CH) - преобразует символ CH в прописной.

Глава 7

Сложный тип данных множество.

Введение в теорию множеств:

Множество - одно из основных понятий современной математики, используемое почти во всех ее разделах.

Понятие множества относится к первоначальным, неопределяемым понятиям математики. Синонимами множества могут быть слова совокупность, класс, собрание и т.п. Можно говорить о множествах книг в библиотеке, учеников класса, фигур на листе бумаги, чисел различной природы и т.д.

Однако согласно теории Георга Кантора одного из основателей теории множества (немецкий математик 1845 - 1918 гг.) любое множество являясь с одной стороны собранием некоторых предметов, в то же время само рассматривается как единое целое, как один предмет: "множество есть многое, мыслимое нами как единое "

Множество X считается заданным (определенным), если относительно любого предмета (объекта) A можно установить, принадлежит ли этот предмет (объект) множеству X или, другими словами, можно определить, будет ли A элементом множества X. Множество задается или перечислением своих элементов или указанием характерных свойств.

Определение:

Множество, не содержащее элементов называется пустым. Пустое множество обозначают: [ ].

Включение и равенство множеств

Определение:

Пусть X и У - два множества. Говорят, что множество X содержится в множестве У, если любой элемент множества X является элементом множества У. Говорят, что множество X является подмножеством множества У или что У включает X .

Определение:

Если одновременно справедливы оба включения, то говорят, что множества равны или совпадают и пишут X=У.

Пример :

Пусть X={1,5,8,7,0} У={0,8,7,1,5} - эти множества равны.

В теории множеств не важен порядок вхождения элементов в множество.

Утверждения:

1. Любое множество является своим подмножеством.
2. Пустое множество содержится в любом множестве.

Операции над множествами

Объединение множеств

Определение

Объединением двух множеств X1 и X2 называется множество, которое состоит из всех элементов множества X1 и X2 и не содержит ни каких других элементов.

Пример:

Пусть X1 = {2,5,7} X2={3,5,6} тогда объединение X1 и X2 ={2,3,5,6,7}

Пересечение множеств

Определение

Пересечением двух множеств X1 и X2 называется такое множество, которое состоит из всех элементов, содержащихся в обоих множествах X1 и X2 , и не содержит никаких других элементов.

Другими словами элемент A принадлежит пересечению множеств X1 и X2 тогда и только тогда, когда A содержится и в X1 и в X2.

Пример:

Пусть X1 = {2,5,7} X2={3,5,6} тогда пересечение X1 и X2 ={5}

Пример:

Пусть X1 - множество целых чисел, делящихся на 2, X2 - множество целых чисел, делящихся на 3, тогда пересечение X1 и X2 - множество целых чисел, делящихся на 6.

Разность множеств

Определение

Разностью двух множеств A и B называется множество A \ B, содержащее те и только те элементы множества A, которые не являются элементами множества B.

Примеры :

1) Пусть X={2,5,7} У={3,5,6} ,тогда X \ У = {2,7}

2) A={2,4,6,8,9} D={6,4,5,9} ,тогда A \ D ={2,8}

Дополнение множеств

Определение

Пусть A подмножество множества X. Множество X \ A называется дополнением A до X и обозначается CxA.

Множественный тип в TP:

Паскаль позволяет оперировать с множествами, как с типами данных. Множества играют важнейшую роль в алгебре. Вполне естественно, что язык Паскаль представляет возможность выполнять операции с множествами.

При описании на языке Паскаль обычно говорят, что множество состоит из элементов, а не объектов. Давая один за другим все элементы, мы тем самым полностью описываем множество.

Например, рассматривая коллекцию марок и указывая все марки из которых состоит коллекция, мы фактически определяем тип всей коллекции.

Множественные типы принадлежат к несколько непривычным и сравнительно редко используемым средствам языка Паскаль. Однако в ряде случаев использование множественных типов позволяет заметно повысить компактность и наглядность программ.

Под множеством в языке Паскаль понимают ограниченный, неупорядоченный набор различных элементов одинакового типа. Всему множеству в целом дается имя. Тип элементов, входящих в множество, называется базовым.

В качестве базового типа можно использовать простые типы: стандартный, кроме действительного, перечисляемый и ограниченный.

Значение множественного типа, так же как и массивы, строятся из нескольких значений одного (базового) типа. Однако, в отличие от массивов и записей , значение множественного типа может содержать любое количество различных элементов базового типа - от ноля элементов (пустое множество) до всех возможных значений базового типа. Иными

словами, возможными значениями переменных множественного типа являются все подмножества значений базового типа.

Описание множественного типа:

Множества должны быть объявлены либо в разделе переменных VAR, либо в разделе типов TYPE. Объявление множества в разделе переменных имеет вид:

var *<имя множества>*: **set of** *<базовый тип>*;

Например:

```
var god:set of 1880..2000;
ch:set of char;
```

Объявление множества в разделе типов имеет вид:

type *<имя множества>* = **set of** *<базовый тип>*;

Например:

```
type
  dayofweek=set of (mon,tue,wed,thu,fri,sat,sun);
  chars=set of char;
  letter=set of 'a'..'z';
  dayofmans=1..31;
var
  d: dayofweek;
  c:chars;
  bookwa:letter;
  day: dayofmans;
```

Необходимо обратить внимание на следующие два обстоятельства: во-первых, все значения базового типа, образующие конкретные значения множественного типа, должны быть различны. Во-вторых порядок "расположения" элементов в множестве никак не фиксируется. Это соответствует принятой в математике трактовке множества как безповторной неупорядоченной совокупности объектов.

Важным является вопрос, из каких значений можно строить множество или, иными словами, каков может быть базовый тип множества. Авторская версия языка ограничивается дискретными типами, однако практически все реализации сильно сужают это ограничение. Так Turbo Pascal допускает в качестве базовых типов для множества дискретные типы не более чем с 256 различными значениями, причем (для целых типов) эти значения должны лежать в диапазоне от 0 до 255. Таким ограничениям удовлетворяют только стандартные типы byte и char,а также перечислимые и ограниченные типы, образованные из них.

Задание множества с помощью конструктора

Значения переменных и констант множества задаются в разделе операторов с помощью конструктора. Конструктор представляет собой список элементов базового типа, заключенных в [ ] скобки.

Например:

```
figura:=[romb,kwadrat,ugol,trg];
bukwa:=['a','b','c'];
ball:=[1..5];
summa:=[ ];  {пустое множество}
```

Таким образом в Паскаль-программе множество задается в виде списка элементов,

заключенного в квадратные скобки. В скобках может быть один или несколько элементов, а может не быть ни одного (пустое множество). В качестве элемента может использоваться константа, переменная, выражение значение которого принадлежит базовому типу, а также паре элементов разделенных двумя точками (интервал значений).

Операции над множествами :

Полезность того или иного типа данных определяется, в первую очередь, набором допустимых операций над значениями этих типов. Что касается множественных типов, то здесь имеются следующие группы операций:

1 - теоретико-множественное объединение, пересечение и вычитание;
+ , * , -

2 - проверка принадлежности элемента множеству;
in

3 - проверка на равенство и неравенство множеств;

4 - проверка на вхождение (принадлежность) одного множества в другое.

Проверка на вхождение в множество:

Эта логическая операция обозначается служебным словом **IN**. Правый операнд должен быть множеством, а левый - значением базового типа множества. Операция вырабатывает true, если значение входит в множество и false, если нет.

Пример:

2 in [1..10,12] вырабатывает true

5 in [1,2,7,10] вырабатывает false.

Операцию проверки принадлежности удобно использовать для исключения более сложных проверок.

Например, оператор вида :

```
if (ch='a') or (ch='b') or (ch='c') or (ch='e') then s
```

удобнее и наглядней применить в виде:

```
if ch in ['a','b','c','e'] then s
```

Кроме того второй вариант эффективнее с точки зрения быстродействия.

Проверка на равенство, неравенство и включение в множество:

Эти бинарные операции также имеют обычный теоретико-множественный смысл и обозначаются следующими знаками :

1. = равенство (совпадение) множеств;
2. <> неравенство множеств;
3. <= проверка на вхождение множества из левого операнда в множество из правого операнда;
4. >= проверка на вхождение множества из правого операнда в множество из левого операнда.

Все эти операции вырабатывают логическое значение true или false в зависимости от успеха проверки.

Пример

```
[1,2,3] = [1,2]     false;
[1,2,3] >=[1,2]     true;
[s] <= [1..10]      true если s целое из диапазона [1,10]
                    false в противном случае;
[1,2,3]<>[1,2,2]    true.
```

Отметим, что все операции над множествами работают достаточно эффективно, поэтому имеет смысл применять их всюду, где это необходимо. К сожалению набор операций над множествами не содержит по крайней мере одной важной операции- выборки значения из множества (или близко связанной с ней – циклического перебора значений множества). Поэтому при необходимости приходится организовывать цикл по всему диапазону значений базового типа, проверяя на каждой итерации принадлежность очередного значения данному множеству, например:

```
Var
  simbol:set of char;
  s:char;
begin
....
  for s:=chhr(0) to chr(255) do
     if s in simbol then     <действия с переменной s>
….
End.
```

Глава 8.

Графические возможности Turbo Pascal.

Известно, что графические образы передают информацию быстрее и нагляднее, чем текст или компьютерные распечатки. В научных и инженерных разработках графическое представление помогает наблюдать за тенденциями изменения, выявлять сложные взаимодействующие факторы и упрощает сопоставление данных. Т.о., использование графики облегчает анализ информации. На персональном компьютере фирмы IBM, оснащенном графическим дисплеем и соответствующей адаптерной платой, можно создавать графические образы быстро и просто.

Графика может быть использована в двух главных направлениях. Во-первых, графика может служить для анализа данных, а во-вторых графика может быть использована для представления результатов анализа данных. Во втором случае результаты исследования представляются в наглядной форме, что облегчает их систематизацию и анализ. Как средство представления информации графика не знает себе равных. Графика способствует повышению качества, позволяет быстро и эффективно создавать выразительные, доходчивые и убедительные изображения.

Графическое изображение на языке Паскаль. Модуль GRAPH

Чтобы сделать процесс графического программирования более эффективным, фирма Borland Int. разработала специализированную библиотеку GRAPH, набор драйверов, позволяющих работать практически со всеми существующими типами мониторов, и набор шрифтов для вывода на графический экран текстов разной величины и формы.

Для формирования графических изображений в языке TP предназначен стандартный библиотечный модуль GRAPH. В нем содержится 79 графических процедур, функций, десятки стандартных констант и типов данных. Все они составляют единый комплекс средств, позволяющих разрабатывать профессиональные программные продукты.

Процедуры и функции модуля Graph:

Все процедуры и функции модуля GRAPH можно разбить на функциональные группы :

1.Управление графическими режимами и их анализом;
2.Рисование графических примитивов и фигур;
3.Управление цветами и шаблонами;
4.Управление выводом текста.

Подключение модуля GRAPH к пользовательской программе осуществляется стандартным способом - с помощью зарезервированного слова USES:

```
Uses Graph;
```

С этого момента все графические средства доступны пользователю.

Драйверы и видеорежимы

Графические драйверы разработаны практически для всех существующих видеоадаптеров, они находятся в так называемых BGI-файлах и активизируются при инициации графического режима.

Драйвер EGAVGA.BGI соответствует адаптеру IBM VGA. Прежде чем работать с графикой, необходимо установить подходящий для имеющегося монитора видеорежим. Для этого не обязательно знать номер видеорежима. Достаточно воспользоваться процедурой

InitGrapf, которая может определить оптимальный режим экрана самостоятельно. Например:

```
Detect:=0;
InitGraph(Detect, A, B);
```

где Detect – переменная, определяющая номер видеорежима,
A – номер подрежима,
B – путь к модулю GRAPH (в нем содержаться процедуры управления графическим экраном). Например, 'C:\TP\GRAPH.TPU'.

Обычно эта процедура устанавливает режим, при котором на экране помещается 640 точек по горизонтали и 480 точек по вертикали, количество цветов – 16 (т.е. режим VGA), что является достаточным для решения большинства задач в школьной программе.

Система координат

Для построения изображений на экране используется система координат. Отсчет начинается от верхнего левого угла экрана, который имеет координаты (0,0). Значение X (столбец) увеличивается слева направо, значение У (строка) - сверху вниз. В режиме монитора VGA графический экран имеет вид:

Группы процедур модуля GRAPH:

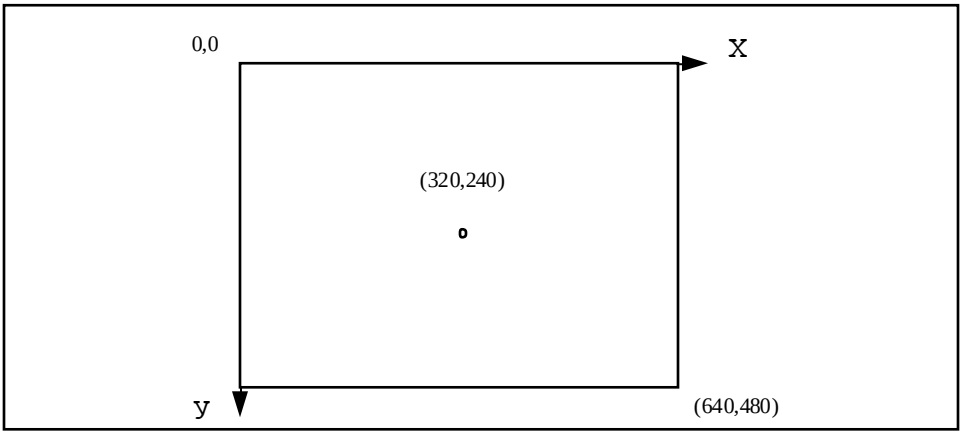
1. Управление графическими режимами и их анализ:

1.1. Инициализация видеорежима - INITGRAPH

Итак начальная группа инструкций должна выглядеть следующим образцом.

В данном примере используется модуль Crt - модуль, содержащий библиотеку процедур, которые работают с клавиатурой и дисплеем, обеспечивая полное управление ими и получение информации об их состоянии.

```
Uses Crt,Graph;
Var
  driver,mode:integer;
begin
```



```
driver:=detect;      {константа detect присваивает значение карты монитора
                      переменной driver и делает автоматическое распознавание
                      монитора}
InitGraph (driver,mode,' '); {процедура инициализации графики}
```

1.2. Закрытие видео режима - CloseGraph

После выполнения всех запланированных графических работ необходимо выйти из графического режима. Это делается с помощью не имеющей параметров процедуры CloseGraph. В процессе выполнения процедура освобождает память, распределенную под драйверы графики, файлы шрифтов и промежуточные данные, и восстанавливает режим работы адаптеров в то состояние, которое было до включения графического режима.

1.3. Очистка графического экрана- ClearDevice

Очистка графического экрана происходит одновременно с установкой графического указателя (аналога курсора в текстовом режиме) в позицию (0,0).

2. Рисование графических примитивов и фигур:

2.1. Управление текущим положением курсором:

Процедуры MovTo, MovRel:

Для построения изображений необходимо указать по крайней мере точку начала вывода. В текстовом режиме эту точку указывает курсор, который присутствует на экране, если его искусственно не убрать. В графических режимах видимого курсора нет, но есть невидимый текущий указатель CP(Current Pointer)

В текстовом режиме (80 на 25) перемещение курсора осуществляется процедурой GoToXY, в графическом режиме для перемещения указателя также имеется ряд процедур и функций . В первую очередь это MovTo и MovRel.

Процедура **MovTo(x,y)** перемещает текущий указатель в точку с координатами X,У.

Например:

MovTo(200,100) перемещает CP в точку экрана с координатами 200,100.

Процедура **MovRel(dX,dY)** перемещает Cp на dX точек по горизонтали и на dY по вертикали.

Например:

MovRel(5,10) перемещает CP из точки с координатами 200,100 в точку с координатами 205,110.

Функции GetX, GetY:

В ряде случаев бывает необходимо выполнять контроль за текущим положением графического указателя. Для этого используют функции GetX и GetY, которые возвращают соответственно значения X и У - координаты CP.

Например:

```
var Xpos,Ypos: integer;
    ...
    x:=GetX; y:=GetY;
    ...
```

Процедуры GetMaxX, GetMaxY:

В процессе управления CP может возникнуть ситуация, когда координаты выходят за рамки экрана, тогда удобно использовать процедуру GetMaxX и GetMaxY, которые воз-

вращают соответственно максимально возможные для установленного режима значения X, У -координат.

Например:

```
...
x:=6000/2; y:=2000/2;
if ((x>GetMaxX) or (y>GetMaxY)) then begin
writeln ( 'Нарушение диапазона');
...
end;
MoveTo(i,j);
...
```

В графических режимах часто приходится определять координаты центра экрана, это можно сделать применив функции GetMaxX, GetMaxY:

```
...
Xcenter:=GetMaxX div 2;
Ycenter:=GetMaxY div 2;
PutPixel(Xcenter,Ycenter,Green); {точка установлена в центре}
...
```

2.2. Рисование графических примитивов:

Процедура вывода точки PutPixel:

Процедура PutPixel (x,y,c) рисует на экране точку с координатами (x, y) цветом с (c=0-15).

Процедура вывода линии - Line:

Процедура line(x1,y1,x2,y2), где x1,y1 - координаты начала линии, x2,y2 - координаты конца линии.

Например:

line(1,1,600,1)- строит горизонтальную прямую в верху экрана.

Процедуры LineTo, LineRel:

LineTo(x,y) - строит линию из точки текущего положения указателя в точку с координатами x,y.

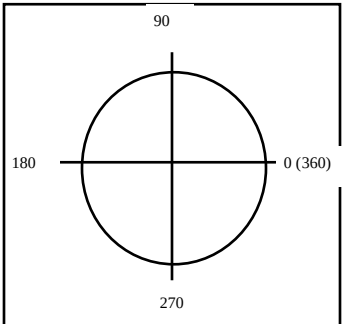


Рис. Полярная система координат

LineRel(dx,dy) - строит линию из точки текущего положения указателя CP в точку с координатами CPx+dx,CPy+dy, где CPx и CPy - текущие координаты указателя CP.

Процедура построения окружностей Circle:

Для задания углов используется полярная система координат. Процедура вычерчивания окружностей текущим цветом имеет формат:

Circle(x,y,Radius:word),

где x,y - координаты центра окружности, radius - радиус;

Например, следующий фрагмент программы позволяет вычерчивать ярко-зеленую окружность с центром в точке 500,100 радиусом 50 пикселов:

```
...
```

```
SetColor(LightGreen);
Circle(500,100,50);
…
```

Процедура рисования прямоугольника Restangle.

Restangle (x1, y1, x2, y2: integer) - рисует не закрашенный прямоугольник, используя текущий цвет и тип линии, x1, y1 – координаты левого верхнего угла прямоугольника, а x2, y2 –координаты нижнего правого угла прямоугольника

Процедуры рисования прямоугольника Bar и трехмерного прямоугольника Bar3D

Bar (x1, y1, x2, y2: integer) - рисует закрашенный прямоугольник (цвет закраски устанавливается процедурой SetFillStyle), используя текущий цвет и тип линии, x1, y1 – координаты левого верхнего угла прямоугольника, а x2, y2 –координаты нижнего правого угла прямоугольника.

Bar3D (x1, y1, x2, y2: integer, h: word, top: boolean) - рисует закрашенный трехмерный прямоугольник, используя текущий цвет и тип линии, x1, y1 – координаты левого верхнего угла прямоугольника, а x2, y2 –координаты нижнего правого угла прямоугольника, h – глубина трехмерного контура, top – указывает, рисовать ли вершину трехмерного контура (top=true – рисовать).

3. Управление цветами и шаблонами

Функция GetPixel (x, y: integer) - определяет цвет точки, которая расположена по указанным координатам (x,y).

Процедура SetColor(c: integer) - устанавливает основного цвета для рисования фигур.

Процедура SetBkColor(c:integer) - устанавливает цвет фона.

Процедура SetFillStyle (Pattern: word, Color: word) - устанавливает образец Pattern и цвет заливки Color.

Процедура SetLineStyle (LineStyle, Pattern, T: word) - устанавливает стиль LineStyle, образец Pattern и толщину линии T.

Процедура FloodFill (x, y: integer, Border: word) - заполняет внутреннюю или внешнюю область фигуры текущим стилем. Здесь x, y – координаты точки внутри закрашиваемой области, Border – цвет границы фигуры.

В следующей таблице приведены константы цветов используемые процедурами **SetBkColor**, **SetColor**, **SetFillStyle**:

Константа	Значение	Константа	Значение
Black	0	DarkGray	8
Blue	1	LightBlue	9
Green	2	LightGreen	10
Cyan	3	LightCyan	11
Red	4	LightRed	12
Magenta	5	LightMagenta	13
Brown	6	Yellow	14
LightGray	7	White	15

Константы типов линий, используемые в процедуре SetLineStyle:

Константа	Значение
SolidLn	0
DottedLn	1
CenterLn	2
DashedLn	3
UserBitLn	4 (Тип линии, определяемый пользователем)
NormWidth	1
ThickWidth	3

Константы прямоугольников, используемые с Bar3D, чтобы указать будет ли 3-хмерная вершина рисоваться на верху прямоугольника:

Константа	Значение
TopOn1	True
TopOff	False

Константы шаблона заполнения, используемые в GetFillSettings и SetFillStyle.

Константа	Значение	
EmptyFill	0	заполняет цветом фона
SolidFill	1	заполняет основным цветом
LineFill	2	--- заполнение
LtSlashFill	3	/// заполнение
SlashFill	4	/// заполнение толстыми линиями
BkSlashFill	5	\\\ заполнение толстыми линиями
LtBkSlashFill	6	\\\ заполнение
HatchFill	7	редкая штриховка
XHatchFill	8	плотная штриховка
InterleaveFill	9	пересекающиеся линии
WideDotFill	10	редкие точки
CloseDotFill	11	плотные точки
UserFill	12	определенный пользователем стиль

4. Управление выводом текста в графическом режиме.

Вывод на экран текста и численных значений в графическом режиме

Выводимые на экран изображения обычно сопровождаются пояснительным текстом. В графическом режиме для этого используются две процедуры.

OutText (TextString: String) - выводит строку текста начиная с текущего положения

графического указателя. Недостаток этой процедуры в том, что нельзя явно указать точку начала вывода. Этого недостатка лишена другая процедура:

OutTextXY (X, Y, Text), где X, Y – координаты точки начала вывода текста.

Пример:

```
OutTextXY (60,120,'Введите данные:');
```

Для начинающих программистов проблемой является вывод числовых данных в графическом режиме, поскольку в модуле GRAPH нет предназначенных для этого процедур. Выход прост: сначала преобразовать число в строку с помощью процедуры Str, а затем воспользоваться процедурой OutTextXY:

```
Max:=34.56;
Str (Max :6 :2, Smax)
OutTextXY (30,200,'Максимум:' + Smax);
```

Остальные процедуры и функции модуля GRAPH приводятся в литературе по Паскалю или в справочной службе интегрированной среды программирования Turbo Pascal.

ПРИЛОЖЕНИЯ.

ПРИЛОЖЕНИЕ А. СООБЩЕНИЯ ОБ ОШИБКАХ.

Сообщения об ошибках компиляции.

Приведенные ниже списки возможных сообщений об ошибках можно получить от компилятора при разработке программы. Когда это возможно, компилятор будет выводить на дисплей дополнительную диагностическую информацию в виде имени идентификатора или файла, например :

Error 15 : File not found (Window.TPU) (Ошибка 15 : Файл не найден)

Когда ошибка обнаружена, Turbo Pascal (в интегрированной среде) автоматически загружает исходный файл и помещает курсор в позицию ошибки. Компилятор командной строки выводит на дисплей сообщение об ошибке и ее номер, а также исходную строку и использует знак вставки (^) для указания места, где произошла ошибка.

Тем не менее заметим, что некоторые ошибки обнаруживаются несколько позже в исходном тексте. Например, несоответствие типов в операторе присваивания не может быть обнаружено до тех пор, пока все выражение после ":=" не будет вычислено. В таких случаях ищите ошибку слева или выше от курсора.

Ниже приводятся наиболее часто встречающиеся на уроках ошибки.

1. Out of memory (не хватает памяти)

Эта ошибка происходит, когда компилятору не хватает памяти. Есть несколько возможных решений этой проблемы:

- если опция Compile/Destination имеет значение Memory, то измените ее значение на Disk в интегрированной среде.

- если опция Options/Compiler /Link buffer в интегрированной среде имеет значение Memory , то измените ее значение на Disk.

2. Identifier expected (ожидается идентификатор)

В этой точке предполагался идентификатор. Возможно, Вы пытаетесь повторно объявить зарезервированное слово.

3. Unknown identifier (неизвестный идентификатор)

Этот идентификатор не объявлен.

4. Duplicate identifier (дублируемый идентификатор)

Этот идентификатор уже был использован в текущем блоке.

5. Syntax error (синтаксическая ошибка)

В исходном тексте был найден неправильный символ. Возможно, Вы забыли кавычки вокруг строковой константы.

6. Error in real constant (ошибка в константе вещественного типа)

7. Error in integer constant (ошибка в константе целого типа)

Заметим, что за целыми числами вещественного типа, лежащими за пределами максимального целого диапазона, должна следовать десятичная точка и ноль; например: 12345678912.0

8. String constant exceeds line (строковая константа выходит за пределы строки). Наиболее вероятно, что Вы забыли закрывающую кавычку в строковой константе.

10. Unexpected end of file (неожиданный конец файла)

Наиболее вероятно, что количество операторов begin и end не сбалансировано.

11. Line too long (строка слишком длинная)

Максимальная длина строки равна 126 символам.

12. Type identifier expected (ожидается тип идентификатора)

Этот идентификатор не обозначает тип, как это должно быть.

13. Too many open files (слишком много открытых файлов)

14. Invalid file name (неправильное имя файла)

15. File not found (файл не найден)

16. Disk full (диск полон) Удалите несколько файлов или используйте новый диск.

20. Variable identifier expected (ожидается идентификатор переменной)

Этот идентификатор не обозначает переменную, как это должно быть.

21. Error in type (ошибка в типе) Этот символ не может начинать определение типа.

23. Set base type out of range (базовый тип множества выходит за допустимый диапазон)

Базовый тип множества должен быть поддиапазоном, который находится внутри диапазона 0..255, или перечислимым типом, который имеет не более 256 возможных значений.

25. Invalid string length (неправильная длина строки)

Объявленная максимальная длина строки должна быть в диапазоне 1..255.

26. Type mismatch (несоответствие типов) Это может быть по следующим причинам:

- несовместимые типы переменной и выражения в операторе присваивания

- несовместимые типы фактического и формального параметра в вызове процедуры или функции

- тип выражения, который несовместим с типом индекса в индексации массива

- несовместимые типы операндов в выражении.

27. Invalid subrange base type (неправильный поддиапазонный базовый тип)

Все порядковые типы являются допустимыми базовыми типами.

28. Lower bound greater than upper bound (нижняя граница больше верхней границы)

29. Orginal type expected (ожидается порядковый тип)

В данном случае вещественный, строковый, структурный тип и тип указателя недопустим.

30. Integer constant expected (ожидается целая константа)

31. Constant expected (ожидается константа)

32. Integer or real constant expected (ожидается целая или вещественная константа)

33. Type identifier expected (ожидается тип идентификатора)

Этот идентификатор не обозначает тип указателя, как это должно быть.

34. Invalid function result type (неправильный тип результата функции)

Правильными типами результатов функций являются все простые типы, строковые типы и типы указателей.

36. Begin expected (ожидается оператор begin)

37. End expected (ожидается оператор end)

38. Integer expession expected (ожидается целое выражение)

39. Ordinal expession expected (ожидается выражение порядкового типа)

40. Boolean expression expected (ожидается выражение булевского типа)

41. Operand types do not match operator (типы операндов не соответствуют оператору). Оператор не может быть применен к операндам этого типа, например 'A' div '2'.

42. Error in expression (ошибка в выражении)

Возможно, Вы забыли написать оператор между двумя операндами.

43. Illegal assignment (неправильное присваивание)

- Файлы и не типизированные переменные не могут быть присваиваемыми значениями.

- Идентификатору функции можно присваивать значения только внутри операторной части функции.

44. Field identifier expected (ожидается идентификатор поля)

Этот идентификатор не обозначает поле в предыдущей переменной типа запись.

50. Do expected (ожидается ключевое слово)

54. OF expected (ожидается ключевое слово OF)

57. THEN expected (ожидается ключевое слово then)

58. TO or DOWNTO expected (ожидается ключевое слово to или downto)

62. Division by zero (деление на ноль). Попытка деления на ноль.

63. Invalid file type (неправильный тип файла)

Этот тип файла не поддерживается процедурой обработки файлов, например, readln с типированным файлом или Seek с текстовым файлом.

64. Cannot Read or Write variables of type (нельзя читать или писать переменные этого типа)

- Read или Readln могут вводить значения типа Char, Integer, Real, и String.

- Write или Writeln могут выводить значения типа Char, Integer, Real, String и Boolean.

66. String variable expected (ожидается строковая переменая)

67. String expession expected (ожидается выражение строкового типа)

69. Unit name mismatch (несоответствие имени модуля)

Имя модуля, найденного в .TPU файле , не соответствует имени, заданному в предложении USES.

74. Constant and case types do not match (константа и тип переключателя в операторе case не соответствуют друг другу)

Тип константы оператора case несовместим с выражением переключателя в предложении case.

75. Record variable expected (ожидается переменная типа record)

76. Constant out of range (константа выходит за допустимый диапазон)

- Попытка индексировать массив с помощью константы, выходящей за допустимый диапазон.

- Попытка присвоить переменной константу, выходящую за допустимый диапазон.

- Попытка передать константу, выходящую за допустимый диапазон, в качестве параметра процедуры или функции.

77. File variable expected (ожидается переменная типа file)

79. Integer or real expression expected (ожидается выражение типа integer или real).

85. ";" expected (ожидается ;)

86. ":" expected (ожидается :)

87. "," expected (ожидается ,)

88. "(" expected (ожидается ( )

89. ")" expected (ожидается))

90. "=" expected (ожидается =)

91. ":=" expected (ожидается :=)

92. "[" or "(." expected (ожидается [или (.)

93. "]" or ".)" expected (ожидается] или .))

94. "." expected (ожидается .)

95. ".." expected (ожидается ..)

97. Invalid FOR control variable (неправильная управляющая переменная в операторе for)

Управляющая переменная в операторе for должна быть простой переменной, определенной в описании текущей подпрограммы.

98. Integer variable expected (ожидается переменная целого типа)

99. File and procedure types are not allowed here (файловый и процедурный тип недопустимы здесь)

Типизированная константа не может быть типа файлового или процедурного типа.

100. String length mismatch (длина строки не соответствует количеству компонент в символьном массиве.

101. Invalid ordering of fields (неправильное упорядочивание полей)

Поля констант типа record должны быть записаны в порядке объявления.

102. String constant expected (ожидается константа типа string)

103. Integer or real variable expected (ожидается переменная типа Integer или Real).104. Ordinal variable expected (ожидается переменная порядкового типа)

106. Character expession expected (ожидается выражение символьного типа)

112. Case constant out of range (константа оператора case выходит за допустимый диапазон). Для операторов case целого типа константы должны быть внутри диапазона -32768...32767.

113. Error in statement (ошибка в операторе). С этого символа не может начинаться оператор.

116. Must be in 8087 mode to compile this (для компиляции этой конструкции должен быть режим 8087)

121. Invalid Qualifier (неправильный квалификатор)

- Попытка индексировать переменную, которая отсутствует в массиве.

- Попытка задать поля в переменной, которая не является записью.

126. Files must be var parameters (файлы должны быть переменными параметрами)

131. Header does not match previous definition (заголовок не соответствует предыдущему определению). Заголовок процедуры или функции, заданный в интерфейсной части или в объявлении forward, не соответствует данному заголовку.

132. Critical dick error (критическая ошибка диска)

134. Expession incorrectly terminated (выражение завершается неправильно)

Turbo Pascal ожидает или оператор, или конец выражения в этой точке, но ничего не находит.

138. Cannot evaluate without System unit (Нельзя вычислить без модуля System)

Библиотека TURBO.TPL должна содержать модуль System для того, чтобы отладчик мог вычислить выражение.

140. Invalid floating-point operation (Неправильная операция с плавающей точкой). Операция над двумя значениями вещественного типа привела к переполнению или является делением на ноль.

142. Procedure or function variable expected (ожидается переменная типа procedure или function)

143. Invalid procedure or function reference (неправильная ссылка на процедуру или функцию)

Ошибки времени выполнения.

Некоторые ошибки во время выполнения программы приводят к тому, что программа выводит на дисплей сообщение об ошибке и завершается:

Run-time error nnn at xxxx:yyyy

(ошибка выполнения nnn по адресу xxxx:yyyy, где nnn-номер ошибки выполнения, а xxxx:yyyy - адрес ошибки выполнения (сегмент и смещение))

Ошибки выполнения разделены на четыре категории:

- ошибки операционной системы DOS:1-99
- ошибки ввода/вывода: 100-149
- критические ошибки : 150-199
- фатальные ошибки: 200-255.

Ошибки операционной системы DOS.

1. Invalid function number (Неверный номер функции)

Вы вызываете несуществующую функцию DOS.

2. File not found (файл не найден)

Сообщается процедурами Reset, Append, Rename или Erase, если имя назначенное файловой переменной не задает существующий файл.

3. Path not fond (путь не найден)

- Сообщается процедурами Reset, Rewrite, Append, Rename или Erase, если имя, назначенное файловой переменной, неправильное или задает несуществующий подкаталог.

- Сообщается процедурами Chdir, Mkdir или Rmdir, если путь доступа неправильный или задает несуществующий подкаталог.

4. Too many open files (слишком много открытых файлов)

Сообщается процедурами Reset, Rewrite или Append, если программа имеет слишком много открытых файлов.

5. File access denied (запрещен доступ к файлу)

6. Invalid file handle (неправильный обработчик файла)

12. Invalid file access code (неправильный код доступа к файлу)

Сообщается процедурами Reset или Append для типизированного или не типизированного файла, если значение FileMode неправильное.

15. Invalid drive number (неправильный номер устройства)

Сообщается процедурой GetDir или ChangeDir.

16. Cannot remove current directory (нельзя удалять текущий каталог)

17. Cannot rename across drives (нельзя переименовывать файлы, находящиеся на различных устройствах). Сообщается процедурой Rename, если оба имени находятся на различных устройствах.

Ошибки ввода/вывода.

100. Disk read error (ошибка чтения с диска)

Сообщается процедурой Read для типизированного файла, если делается попытка чтения после конца файла.

101. Disk write error (ошибка записи на диск)

Сообщается процедурами Close, Write, Writeln, Flush или Page, если диск становится полным.

102. File not assigned (файл не назначен)

Сообщается процедурами Reset, Rewrite, Append, Rename или Erase, если файловой переменной не было присвоено имя с помощью вызова Assign.

103. File not open (файл не открыт). Сообщается процедурами Close, Read, Write, Eof, FileSize, Flush, BlockRead или BlockWrite, если файл не открыт.

104. File not open for input (файл не открыт для ввода). Сообщается процедурами Read, Readln, Eof, Eoln, SeekEof, SeekEoln для текстовых файлов, если файл не открыт для ввода.

105. File not open for output (файл не открыт для вывода). Сообщается процедурами Write и Writeln для текстовых файлов, если файл не открыт для вывода.

106. Invalid numeric format (неправильный числовой формат)

Сообщается процедурами Read и Readln, если числовое значение, прочитанное из текстового файла не соответствует правильному числовому формату.

Критические ошибки.

150. Disk is write-protected (защита диска от записи)
151. Unknown unit (неизвестное устройство)
152. Drive not ready (устройство не готово)
153. Unknown command (неизвестная команда)
154. CRC error in data (ошибка циклического контроля по избыточности в данных)
155. Bad drive reguest structure length (неправильный запрос устройства о длине структуры)
156. Disk seek error (ошибка поиска на диске)
157. Unknown media type (неизвестный тип носителя)
158. Sector not found (сектор не найден)
159. Printer out of paper (на принтере нет бумаги)
160. Device write fault (ошибка записи на устройство)
161. Device read fault (ошибка чтения с устройства)
162. Hardware failure (отказ аппаратного обеспечения)

Фатальные ошибки.

Эти ошибки всегда сразу же завершают программу.

200. Division by zero (деление на ноль)

201. Ошибка выхода за допустимый диапазон. Эта ошибка сообщается операторами, когда возникает одна из следующих ситуаций:

- Индексное выражение квалификатора массива выходит за допустимый диапазон.
- Была сделана попытка присвоить переменной значение, выходящее за допустимый диапазон.
- Была сделана попытка передать процедуре или функции в качестве параметра значение, выходящее за допустимый диапазон.

202. Stack overflow error (ошибка переполнения стека)

Эта ошибка сообщается при входе в процедуру или функцию, когда в стеке нет достаточно места для размещения локальных переменных подпрограмм. Увеличьте размер стека.

Эта ошибка может возникать так же из-за бесконечной рекурсии.

205. Floating point overflow (переполнение с плавающей точкой). Операция с плавающей точкой обрабатывает слишком большое для Turbo Pascal или сопроцессора число.

207. Invalid floating point operation (неправильная операция с плавающей точкой)

- Вещественное значение, передаваемое Trunc или Round, не может быть преобразовано в целое внутри диапазона для типа LongInt (-2147483648 до 214783647)
- Аргумент, передаваемый функции Sgrt, является отрицательным числом.
- Аргумент, передаваемый функции Ln, равен нулю или отрицательный.
- Произошло переполнение стека сопроцессора 8087.

ПРИЛОЖЕНИЕ В

Специальные символы и зарезервированные слова

Turbo Pascal использует следующий набор символов кода ASCII:
- Буквы английского алфавита от A до Z и от a до z;
- Цифры - арабские от 0 до 9;
- Шестнадцатеричные цифры - арабские цифры от 0 до 9, буквы от A до F и от a до f.
- Пробелы - символ пробела (ASCII 32) и все управляющие символы кода ASCII (ASCII от 0 до 31), включая символ конца строки или символ возврата каретки (ASCII 13).

Специальные символы и зарезервированные слова представляют собой символы, имеющие одно или несколько фиксированных значений.
Специальными символами являются следующие символы:

+ - * / = < > [ ] . , ( ) : ; ^ @ { } $ #

Следующие пары символов также представляют собой специальные символы:

<= >= := .. (* *) (. .)

Некоторые специальные символы являются также операторами. Левая квадратная скобка ([) эквивалентна паре символов, состоящей из левой круглой скобки и точки ((.). Аналогично правая квадратная скобка (]) эквивалентна паре символов, состоящей из точки и правой круглой скобки (.)).

Следующие слова являются зарезервированными в Турбо-Паскале:

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

Для Turbo Pascal безразличен регистр клавиатуры, поэтому Вы можете использовать в своей программе как строчные, так и прописные буквы.

Идентификаторы

Идентификаторы выступают в качестве имен констант, типов, переменных, процедур, функций, модулей, программ и полей в записях. Идентификатор может иметь любую длину, однако значимыми являются только первые 63 символа.

Идентификатор должен начинаться с буквы или символа подчеркивания и не должен содержать пробелов. После первого символа допускаются буквы, цифры и знак подчеркивания (ASCII $5F). Как и зарезервированные слова, идентификаторы безразличны к регистру клавиатуры.

Числа

Для чисел, представляющих собой константы целого и вещественного типа, используется обычная десятичная запись. Целая константа в шестнадцатиричном формате имеет в качестве префикса знак доллара ($). Техническое обозначение (E или e с экспонентой) в вещественных типах чи-

тается как "десять в степени".

Например:

```
7e-2        "7 умноженное на 10 в степени -2";
12.25e+6    "12.25 умноженное на 10 в степени +6";
```

Строки символов

Строка символов представляет собой последовательность, содержащую нуль и более символов из расширенного набора символов кода ASCII (приложение В), записанную в одной строке программы и заключенную в одиночные кавычки (апострофы). Строка символов, ничего не содержащая между апострофами, называется пустой строкой. Два последовательных апострофа в строке символов обозначают один символ - апостроф. Атрибут длины строки символов содержит действительное количество символов между апострофами.

В качестве расширения стандартного Паскаля Turbo Pascal разрешает вставлять в строку символов управляющие символы. Символ # с целой константой без знака в диапазоне от 0 до 255 обозначает соответствующий этому значению символ в коде ASCII. Между символом # и целой константой не должно быть никаких разделителей. Аналогично, если несколько управляющих символов входят в строку символов, то между ними не должно быть разделителей.

Строки программы

В Turbo Pascal строки программы имеют максимальную длину в 126 символов.

ПРИЛОЖЕНИЕ С

Задачник по программированию в среде Turbo Pascal

I. Задачи по теме « Программирование числовых последовательностей»

Общее условие для всех задач: вычислить сумму (произведение) элементов числовых последовательностей.

1. Вычислить значение выражения состоящего из n элементов, n ввести с клавиатуры:
1.1. y=1+3+5+7+11…
1.2. y=n! (факториал)
1.3. y=1/2+3/5+5/8+7/11+…
1.4. y=(1/2+2/3+3/4+...)/(2/5+4/10+8/15+...)
1.5. y=(1+2+4+8+...)/(5+7+9+11+..)
1.6. y=(3*5+5*7+7*9+9*11+...)/n!
1.7. y=1/2!+1/3!+1/4!+...
1.8. y=(2+4+8+16+...)/n!

2. Вычислить значение выражения состоящего из n элементов. Значение «x» ввести с клавиатуры.
2.1. y=2*x+5*x+8*x+11*x+...
2.2. y=(x*x*...*x)/n! (факториал)
2.3. y=x/2+(3*x)/5+(5*x)/8+(7*x)/11+...
2.4. y=(x/2+2*x/3+3*x/4+...)/(2*x/5+4*x/10+8*x/15+...)
2.5. y=(1*x+2*x+4*x+8*x+...)/n!
2.6. y=(3*5*x+5*7*x+7*9*x+9*11*x+...)/n!
2.7. y=x/2!+x/3!+x/4!+...
2.8. y=(x+2*x+3*x+5*x+7*x+...)/n!

3. Вычислить значение выражения с точностью, значение которой задается с клавиатуры. Значение "x" ввести с клавиатуры.
3.1. y=x*1/2+x*2/3+x*3/4+x*5/6+x*6/7+...
3.2. exp(x)=1+x/1!+x*x/2!+x*x*x/3!+x*x*x*x/4!+...
3.3. sh(x)=x+x*x*x/3!+x*x*x*x*x/5!+x*x*x*x*x*x/6!+...
3.4. ch(x)=1+x*x/2!+x*x*x*x/4!+x*x*x*x*x*x/5!+...
3.5. sin(x)=x-x*x*x/3!+x*x*x*x*x/5!-x*x*x*x*x*x*x/7!+...

Примеры решения задач

```
Program Posledov;
Uses Crt;
Var n, i: Integer;
  x, t, y, p, z: Real;
Begin
  ClrScr;
{Ввод исходных данных}
  Write ('Введите n:'); Readln (n);
  Write ('Введите x:'); Readln (x);
{————————————————————————————————————————
1.1. Вычислить сумму элементов последовательности: y=1+3+5+7+11…
————————————————————————————————————————}
  y:=0;
```

```
  For i:=1 to n do
  y:=y+i*2-1;
  Writeln ('Сумма элементов последовательности:', y);
{-----------------------------------------
1.2. Вычислить факториал n:   y=n!
-----------------------------------------}
  y:=1;
  For i:=1 to n do
  y:=y*i;
  Writeln ('Факториал числа n:', y);
{-----------------------------------------------------
2.1. Вычислить сумму элементов последовательности:
                y=2*x+5*x+8*x+11*x+...
-----------------------------------------------------}
  y:=0;
  For i:=1 to n do
  y:=y+(i*3-1)*x;
  Writeln ('Сумма элементов последовательности:', y);
{-----------------------------------------------------
2.2. Вычислить произведение элементов последовательности:
                      y=(x*x*...*x)/n!
-----------------------------------------------------}
  y:=1; p:=1;
  For i:=1 to n do
  Begin
  p:=p*x;
  y:=y*i;
  Writeln ('Произведение элементов последовательности:', p/y);
{-----------------------------------------------------
3.5. Вычислить сумму элементов последовательности:
        sin(x)=x-x*x*x/3!+x*x*x*x*x/5!-x*x*x*x*x*x*x/7!+...
-----------------------------------------------------}
  y:=0;  p:=0;      {присваивание начальных значений переменным: i – счетчик
  x:=pi;  i:=1;циклов; p, y – числитель и знаменатель элементов}
  t:=0.00001;       {точность вычисления суммы}
  Flag:=True;       {переменная для определения знака слагаемого}
Repeat              {начало цикла для подсчета суммы последовательности}
  p:=p*x;   z:=z*i; {вычисление числителя и знаменателя элемента}
  If Flag Then      {проверка значения флага}
  Begin             {если флаг – истина, то элемент прибавляется}
    y:=y+p/z;       {вычисление значения элемента и накопление суммы}
    Flag:=False;    {изменения знака для следующего элемента}
  End
  Else              {ветвь, которая выполняется, если флаг – ложь}
  Begin             {если флаг – ложь, то элемент вычитается}
    y:=y-p/z;       {вычисление значения элемента и накопление суммы}
    Flag:=True;     {изменения знака для следующего элемента}
  End;              {конец ветвления}
  i:=i+1;           {увеличения на 1 значения счетчика элементов}
```

```
Until  t > abs ((y-sin(x))); {конец цикла Repeat и проверка условия окончания
                              цикла}
  Writeln ('i=',i);   {вывод результатов на экран}
  Writeln ('      t=',t:12:9);
  Writeln ('      y=',y:12:9);
  Writeln ('sin(x)=',sin(x):12:9);
Readln;               {задержка результатов на экране до нажатия Enter}
End.
```

II. Задачи по теме «Ввод и обработка числовой информации в однопроходных алгоритмах»

Требования к аттестации.

В начале изучения темы преподаватель назначает учащемуся номер его варианта. Стандартные задачи разбираются у доски, комбинированные задачи учащийся решает самостоятельно. К итоговой аттестации по теме в тетради для домашних и самостоятельных работ у учащегося должно быть решено дома не менее 3 задач с блок схемами. В классе учащийся должен отладить на ЭВМ не менее 3 задач, предлагаемых для самостоятельной работы в классе.

На итоговой контрольной необходимо решить 3 задачи. На 3 - N1 (стандартная задача, комментарии и блок схема к ней), на 4 - любую пару задач, на 5 - все три. Для одной из задач (любой) должна быть изображена блок схема алгоритма и к каждой строке должны быть написаны подробные комментарии. Также должна быть предъявлена дискета с задачами отлаженными в классе.

Общая часть для всех задач:

Вводить с клавиатуры n раз (n-вводится с клавиатуры) в память ЭВМ целые числа и при этом производить их обработку, согласно конкретному условию данной задачи. При решении задачи массивы не использовать!

БЛОК 1

Стандартные алгоритмы.

1.1. Вычислить среднеарифметическое значение введенных чисел.

1.2. Определить количество введенных положительных чисел.

1.3. Вычислить среднеарифметическое значение введенных положительных чисел.

1.4. Вычислить среднеарифметическое значение введенных четных чисел.

1.5. Вывести на экран удвоенное значение последнего введенного четного числа.

Самостоятельная работа №1.

1.6. (Четные ЭВМ) Вычислить сумму 1)среднеарифметического значения чисел кратных трем и 2) последнего введенного нечетного числа.

1.7 (Нечетные ЭВМ) Вычислить произведение 1)количества чисел кратных пяти и 2)последнего введенного четного числа.

Домашнее задание №1.

1.8 Вычислить сумму последнего введенного числа кратного трем и 2)среднеарифметического значения всех нечетных чисел.

1.9 Найти разность количества четных и нечетных чисел.

1.10 Вычислить произведение между последним числом кратным пяти и суммой всех нечетных чисел.

1.11 Вычислить произведение между последним числом кратным пяти и среднеарифметическим значением всех четных чисел.

1.12 Вычислить сумму между количеством нечетных чисел и последним введенным числом.

1.13 Вычислить произведение между последним числом кратным девяти и суммой всех введенных чисел.

1.14 Вычислить сумму между последним числом кратным семи и среднеарифметическим значением всех нечетных чисел.

1.15 Вычислить разность между количеством четных чисел и среднеарифметическим значением нечетных чисел.

1.16 Вычислить разность между последним числом кратным шести и суммой всех чисел кратных трем.

БЛОК 2

Стандартные алгоритмы.

2.1 Вывести на экран значение первого введенного четного числа.

2.2 Вывести на экран значение произведения первого и последнего введенного четного числа.

2.3 Определить значение самого минимального введенного числа.

2.4 Определить значение номера самого минимального введенного числа.

Самостоятельная работа №2.

2.5 (Четные ЭВМ) Вычислить среднеарифметическим значение между первым числом кратным трем и максимальным введенным числом.

2.6 (Нечетные ЭВМ) Вычислить произведение между количеством чисел кратных пяти и максимальным введенным числом.

Домашнее задание №2

2.7 Вычислить сумму минимального числа со среднеарифметическим значением между первым и последним нечетным числом.

2.8 Вычислить разность между количеством четных чисел и первым нечетным числом.

2.9 Вычислить произведение между максимальным числом и первым среди нечетных чисел.

2.10 Вычислить произведение между первым введенным числом кратным пяти и минимальным значением среди всех чисел.

2.11 Вычислить сумму между количеством нечетных чисел и первым введенным числом.

2.12 Вычислить произведение между первым числом кратным девяти и минимальным среди всех введенных чисел.

2.13 Вычислить сумму между первым введенным числом кратным семи и максимальным среди всех чисел.

2.14 Вычислить разность между количеством четных чисел и минимальным среди всех чисел.

2.15 Вычислить разность между первым числом кратным шести и максимальным числом среди всех чисел.

БЛОК 3

Стандартные алгоритмы.

3.1 Найти максимальное нечетное число.

3.2 Найти максимальное число среди последних трех введенных чисел.

Самостоятельная работа №3.

3.3 (Четные ЭВМ) Вычислить произведение между первым числом кратным трем и минимальным введенным нечетным числом.

3.4 (Нечетные ЭВМ) Вычислить сумму между количеством чисел кратных пяти и максимальным введенным четным числом.

Примеры решения задач.

```
{Общая часть}
Program read_write;
  uses crt;
  const n=5;
  var i, x, s, k, max, min, nmax, nmin:integer;
begin
{ Задача 1.1/ Вычислить среднеарифметическое значение введенных чисел.}
  clrScr;
```

```
  s:=0;              {обнуление счетчика}
  for i:=1 to n do
    begin
      write ('x',i,'=');    {вывод комментария}
      readLn (x);           {ввод числа в переменную "x"}
      s:=s+x;               {счетчик для вычисления суммы чисел }
    end;
  write ('1.1 Ответ =',s/n:4:2);
  repeat until keyPressed;
```

{ Задача 1.2. Определить количество введенных положительных чисел.}

```
  clrscr;
  k:=0;      {обнуление счетчика}
  for i:=1 to n do
    begin
      write ('x',i,'=');
      readLn (x);
      if x>0 then  k:=k+1;     {вычисление количества положительных чисел}
    end;
  write ('1.2. Ответ =',k);
repeat until keyPressed;
```

{ Задача 1.3. Вычислить среднеарифметическое значение введенных положительных чисел.}

```
  clrscr;
  s:=0;      {обнуление счетчика}
  k:=0;      {обнуление счетчика}
  for i:=1 to n do
    begin
      write ('x',i,'=');
      readLn(x);
      if x>0 then
        begin
          k:=k+1; {счетчик для вычисления количества положительных чисел}
          s:=s+x; {счетчик для вычисления суммы положительных чисел}
        end;
    end;
  write ('1.3 Ответ =',s/k:4:2);
  repeat until keyPressed;
```

{ Задача 1.4 Вычислить среднеарифметическое значение четных чисел.}

```
  clrscr;
  s:=0;      {обнуление счетчика}
  k:=0;      {обнуление счетчика}
  for i:=1 to n do
    begin
      write ('x',i,'=');
      readLn(x);
      if x/2=int(x/2) then       {проверка на четность}
```

```
        begin
          k:=k+1;          {счетчик для вычисления количества четных чисел}
          s:=s+x; {счетчик для вычисления суммы четных чисел}
        end;
    end;
    if k<>0 then write ('1.4 Ответ =',s/k:4:2) else write ('Четных чисел нет!');
  repeat until keyPressed;

{ Задача 1.5  Вывести на экран удвоенное значение последнего введенного четного числа.}
  clrscr;
  k:=666; {"метка" помещается в переменную, в которую будем класть
           четные числа}
  for i:=1 to n do
    begin
      write ('x',i,'=');
      readLn(x);
      if x/2=int(x/2) then        {проверка на четность}
        begin
          k:=x;     {помещаем в "k" найденное четное число}
        end;
    end;
  if k<>666 then write('1.5 Ответ =',k) else write('Четных чисел нет!');
  repeat until keyPressed;

{ 2.1. Вывести на экран значение первого введенного четного числа.}
{1-й вариант с циклом "for"}
  clrscr;
  k:=666; {"метка" помещается в переменную, в которой будем хранить  четные
      числа}
  for i:=1 to n do
    begin
      write ('x',i,'=');
      readLn(x);
      if (x/2=int(x/2)) and (k<>666) then {проверка на четность}
        begin
          k:=x;     {помещаем в "k" найденное четное число}
        end;
    end;
  if k<>666 then write ('2.1 Ответ =',k) else write ('Четных чисел нет!');
  repeat until keyPressed;
  clrscr;
{2-й вариант с циклом "repeat"}
  i:=1;
  repeat
    write ('x',i,'=');
    readLn(x);
    i:=i+1;
  until (x/2=int(x/2)) or (i=n+1);     {проверка на четность и выход из цикла}
  if x/2=int(x/2) then write ('2.1 Ответ =',x) else write ('Четных чисел нет!');
```

```
repeat until keyPressed;

{2.2. Вывести на экран значение произведения первого и последнего введенного четного числа.}
  k:=1;
  repeat
    write ('x',k,'=');
    readLn(x);
    k:=k+1;
  until (x/2=int(x/2)) or (k=n+1);  {проверка на четность и выход из цикла}
  y1:=x;
  if k>n then
    Begin
      For i:=k to n do
      Begin
        write ('x',i,'=');
        readLn (x);
        if x/2=int (x/2) then  y2:=x;
      End;
    Write ('2.2. Ответ:', y1*y2)
  else write ('Четных чисел нет!');
repeat until keyPressed;

{ 2.3. Определить значение самого минимального введенного числа.}
  write ('x',i,'=');
  readLn (x);
  Min:=x;   {начальное значение переменной min}
  For i:=2 to n do
    Begin
      write ('x',i,'=');
      readLn (x);
      if x<min then  min:=x;   {присваивание нового значения переменной min}
    End;
  Write ('2.3. Ответ:', min);
  repeat until keyPressed;

{ 2.4 Определить значение номера самого минимального введенного числа.}
  write ('x1=');
  readLn (x);
  Min:=x;
  nmin:=1
  For i:=2 to n do
    Begin
      write ('x',i,'=');
      readLn (x);
      if x<min then
      Begin
        min:=x;          {присваивание нового значения переменной min}
        nmin:=i;         {присваивание нового значения переменной nmin}
```

```
      End
    End;
  Write ('2.4. Ответ:', nmin)
  repeat until keyPressed;

{ 3.1 Найти максимальное нечетное число.}
  write ('x1=');
  readLn (x);
  Max:=x;
  For i:=2 to n do
    Begin
      write ('x',i,'=');
      readLn (x);
      if (x<max) and (x/2<>int(x/2)) then
        max:=x;          {присваивание нового значения переменной max}
    End;
  if x/2<>int(x/2) then write ('3.1 Ответ =',max) else write ('Нечетных чисел нет!');
  repeat until keyPressed;

 3.2 Найти максимальное число среди последних трех введенных чисел.}
  For i:=1 to n do
    Begin
      write ('x',i,'=');
      readLn (x);
      If  i=n-2 Then Max:= x;
      If  (i>(n-2)) and (x<max) Then
        max:=x; {присваивание нового значения переменной max}
    End;
  ('3.2 Ответ =',max);
  repeat until keyPressed;
End.
```

III. Задачи по теме «Одномерные массивы переменных»

На итоговой контрольной необходимо решить 3 задачи. На 3 - одну задачу (стандартная задача, комментарии и блок схема к ней), на 4 - любую пару задач, на 5 – все три. Для любой из задач должна быть изображена блок схема алгоритма и к каждой строке должны быть написаны подробные комментарии. Также должна быть предъявлена дискета с задачами отлаженными в классе.

На зачете или экзамене, кроме решения задачи, необходимо ответить письменно или устно на контрольные вопросы указанные в билете, а также объяснить подробно один из стандартных алгоритмов.

P.S. Те учащиеся для которых уровень предлагаемых задач слишком низок могут решить три задачи повышенной сложности, а затем задачи из районных олимпиад по программированию.

Общее условие для всех задач:

Создать в памяти ЭВМ одномерный массив из n-целых чисел (n - константа) с помощью генератора случайных чисел, по формуле или задавая значения элементов массива

с клавиатуры.

После запуска созданной Вами программы на экран должны быть выведены исходные значения всех элементов массива и результат его обработки.

БЛОК 1.

Стандартные алгоритмы.

1.1. Вычислить сумму величин элементов массива.

1.2. Вычислить среднее арифметическое всех нечетных (четных) величин элементов массива.

1.3. Найти максимальное (минимальное) значение из всех величин элементов массива.

1.4. Найти номер элемента массива, величина которого максимальна (минимальна).

Самостоятельная работа №1 на ЭВМ.

1.5. (1, 5 ЭВМ) Вычислить разность между максимальным элементом массива и величиной первого элемента массива.

1.6. (2, 6 ЭВМ) Вычислить сумму между минимальным элементом массива и суммой всех его нечетных элементов.

1.7. (3, 7, 9 ЭВМ) Вычислить сумму между минимальным элементом массива и величиной последнего элемента массива.

1.8. (4, 8, 10 ЭВМ) Вычислить разность между минимальным элементом массива и суммой всех его четных элементов.

Домашнее задание №1.

1.9. Вычислить среднее арифметическое величин первых трех элементов массива кратных двум.

1.10. Вычислить разность между максимальным и минимальным элементами массива.

1.11. Вывести на экран значения всех элементов массива которые расположены до минимального элемента массива.

1.12. Вывести на экран значения всех элементов массива которые расположены после минимального элемента массива.

1.13. Найти максимальное значение из среди всех элементов массива которые расположены до минимального элемента.

1.14. Вывести на экран значения всех элементов массива которые расположены между минимальным и максимальным элементами массива.

1.15. Вычислить разность между произведением всех четных элементов массива и их среднеарифметическим значение.

1.16. Вывести на экран сообщение "Максимум в массива стоит перед минимумом" или сообщение противоположного смысла в зависимости от конкретного содержания массива.

1.17. Вычислить среднее арифметическое величин последних трех элементов массива кратных двум.

1.18 Вычислить произведение между максимальным и минимальным элементами массива.

1.19. Вывести на экран сумму значений тех элементов массива которые расположены до минимального элемента массива.

1.20. Вывести на экран среднее арифметическое тех элементов массива которые расположены после минимального элемента массива.

1.21. Найти минимальное значение из среди тех элементов массива которые расположены после максимального элемента.

1.22. Вывести на экран сумму тех элементов массив которые расположены между минимальным и максимальным элементами массива.

1.23. Вычислить сумму между произведением всех четных элементов массива и их среднеарифметическим значение.

1.24. Вывести на экран сообщение "Минимум находится в массива перед максимумом" или сообщение противоположного смысла в зависимости от конкретного содержания массива.

БЛОК N 2.

Стандартные алгоритмы.

2.1. Найти минимальный нечетный элемент массива.

2.2. Поменять местами минимальный элемент массива с максимальным.

2.3. Поменять местами попарно-равноудаленные от краев массива элементы (1 и n, 2 и n-1,3 и n-2

и т.д.).

2.4 Поменять попарно-соседние элементы массива (1 и 2, 3 и 4, и т.д.).

Самостоятельная работа №2 на ЭВМ.

2.5. (1 ЭВМ) Поменять местами минимальный четный элемент массива с просто максимальным.

2.6. (2 ЭВМ) Поменять местами минимальный положительный элемент массива с крайним левым отрицательным элементом массива.

2.7. (3 ЭВМ) Поменять местами максимальный нечетный элемент массива с просто минимальным.

2.8. (4 и 9 ЭВМ) Поменять местами максимальный отрицательный элемент массива с крайним правым отрицательным элементом массива.

2.9. (5 и 10 ЭВМ) Поменять местами минимальный элемент массива с максимальным нечетным.

2.10. (8 ЭВМ) Поменять местами минимальный отрицательный элемент массива с крайним правым положительным элементом массива.

2.11. (7 ЭВМ) Поменять местами максимальный элемент массива с минимальным нечетным.

2.12. (8 ЭВМ) Поменять местами минимальный кратный трем элемент массива с крайним левым четным элементом массива.

Домашнее задание №2.

2.13. Найти минимальный нечетный элемент и поменять его с зеркально расположенным элементом массива.

2.14. Найти максимальный элемент кратный трем и увеличить его значение на величину находящегося справа от него элемента.

2.15. Найти максимальный отрицательный элемент массива и вычесть его значение от всех положительных элементов.

2.16. Найти минимальный элемент массива кратный пяти и поменять его местами с первым элементом массива.

2.17. Найти максимальный нечетный элемент и поменять его с зеркально расположенным элементом массива.

2.18. Найти минимальный элемент кратный двум и увеличить его значение на величину находящегося справа от него элемента.

2.19. Найти минимальный положительный элемент массива и вычесть его значение от всех положительных элементов.

2.20. Найти максимальный элемент массива кратный трем и поменять его местами с последним элементом массива.

2.21. Найти минимальный четный элемент и поменять его с максимальным четным элементом массива.

2.22. Найти максимальный элемент кратный трем и увеличить его значение на величину находящегося справа от него элемента.

2.23. Найти максимальный отрицательный элемент массива и увеличить его значение на значение минимального положительного элемента.

2.24. Найти минимальный элемент массива кратный трем и поменять его местами с первым четным элементом массива.

2.25. Найти максимальный кратный трем элемент и поменять его с зеркально расположенным элементом массива.

2.26. Найти минимальный элемент кратный пяти и увеличить его значение на величину находящегося справа от него первого четного элемента.

2.27. Найти минимальный положительный элемент массива и вычесть его значение от всех элементов массива кратных пяти.

2.28. Найти максимальный элемент массива кратный трем и поменять его местами с последним элементом массива не кратным трех.

БЛОК 3

Стандартные алгоритмы.

3.1 Сдвинуть все элементы массива на k-позиций вправо. Значение "k" задается с клавиатуры.

3.2 Рассортировать все элементы массива по возрастанию методом перестановок (метод

"пузырька").

Самостоятельная работа №3 на ЭВМ.

3.3 (1,5 ЭВМ) Сдвинуть все элементы массива вправо таким образом, чтобы минимальный четный элемент массива оказался на последнем месте.

3.4 (2,6 ЭВМ) Рассортировать все элементы массива находящиеся между минимальным нечетным и максимальным четным элементами массива.

3.5 (3,7,9 ЭВМ) Сдвинуть все элементы массива влево таким образом, чтобы максимальный нечетный элемент массива оказался на первом месте.

3.6 (4,8,10 ЭВМ) Рассортировать все элементы массива находящиеся между максимальным четным и минимальным нечетным элементами массива.

Домашнее задание №3.

3.7. Скопировать все элементы массива величина которых четное число в дополнительный массив "Y" и рассортировать их там по убыванию.

3.8. Скопировать все элементы массива находящиеся после максимального четного числа в в дополнительный массив "Y" и рассортировать их там по возрастанию.

3.9. Скопировать все элементы массива меньшие максимального четного числа в дополнительный массив "Y" и рассортировать их там по убыванию.

3.10. Скопировать все элементы массива меньшие среднеарифметического значения величин всех его элементов в дополнительный массив "Y" и рассортировать их там по возрастанию.

3.11. Скопировать все элементы массива величина которых нечетное число в дополнительный массив "Y" и рассортировать их там по возрастанию.

3.12. Скопировать все элементы массива находящиеся после минимального четного числа в в дополнительный массив "Y" и рассортировать их там по убыванию.

3.13. Скопировать все элементы массива меньшие максимального нечетного числа в дополнительный массив "Y" и рассортировать их там по возрастанию.

3.14. Скопировать все элементы массива большие среднеарифметического значения величин всех его элементов в дополнительный массив "Y" и рассортировать их там по убыванию.

3.15. Скопировать все элементы массива величина которых кратна трем в дополнительный массив "Y" и рассортировать их там по убыванию.

3.16. Скопировать все элементы массива находящиеся между максимальным четным числом и первым элементом массива в в дополнительный массив "Y" и рассортировать их там по возрастанию.

3.17. Скопировать все элементы массива большие максимального числа кратного трем в в дополнительный массив "Y" и рассортировать их там по убыванию.

3.18. Скопировать все элементы массива меньшие среднеарифметического значения величин всех его четных элементов в дополнительный массив "Y" и рассортировать их там по возрастанию.

3.19. Скопировать все элементы массива величина которых больше среднеарифметического величин всех четных элементов в дополнительный массив "Y" и рассортировать их там по убыванию.

3.20. Скопировать все элементы массива находящиеся до максимального числа кратного трем в дополнительный массив "Y" и рассортировать их там по возрастанию.

3.21. Скопировать все элементы массива меньшие максимального четного числа в дополнительный массив "Y" и рассортировать их там по убыванию.

3.22. Скопировать все элементы массива меньшие среднеарифметического значения величин всех его нечетных элементов в дополнительный массив "Y" и рассортировать их там по возрастанию.

БЛОК №4

Стандартные алгоритмы.

4.1. Рассортировать все элементы массива по возрастанию методом выбора.

4.2. Скопировать все четные элементы массива расположенные между минимальным четным элементом и максимальным нечетным элементом в дополнительный массив "Y" таким образом чтобы они располагались в нем подряд с левого края.

На свободные места записать нули. Например:

Массив "X": 1,9,6,4,5,1,8,2,4,7

Массив "Y": 6,4,8,0,0,0,0,0

Самостоятельная работа №4 на ЭВМ.

4.3. (1,5 ЭВМ) В дополнительный массив "Y" записать начиная с левого края суммы элементов

исходного массива расположенные симметрично от его краев. Например: Массив "X": 1,9,6,4,5

Массив "Y": 6,13,6

Рассортировать все элементы массива "Y" по возрастанию методом выбора.

4.4. (2,6 ЭВМ) В дополнительный массив "Y" записать начиная с левого края суммы элементов исходного массива индексы которых расположены рядом. Например:

Массив "X": 1,9,6,4,5

Массив "Y": 13,15,10,9

Рассортировать все элементы массива "Y" по возрастанию методом выбора.

4.5. (3,7,9 ЭВМ) В дополнительный массив "Y" записать начиная с левого края суммы троек элементов исходного массива индексы которых расположены рядом.

Например:

Массив "X": 1,9,6,4,8,4,5

Массив "Y": 16,19,18,16,17

Рассортиривать все элементы массива "Y" по возрастанию методом выбора.

4.6. (4,8,10 ЭВМ) В дополнительный массив "Y" записать начиная с левого края суммы пар элементов исходного массива индексы которых расположены рядом.

Например:

Массив "X": 1,9,6,4,8,4,5

Массив "Y": 13,10,12,5

Рассортировать все элементы массива "Y" по возрастанию методом выбора.

Задачи повышенной сложности.

5.1. "Модель лототрона 1". Сформировать массив размерностью "n", заполненный натуральными числами от 1 до "n" (следовательно числа в массиве не повторяются). Числа должны быть расположены в массиве случайным образом.

ОДИН ИЗ ВАРИАНТОВ РЕШЕНИЯ: Заполнить массив натуральными числами равными индексу этого элемента массива. "Перемешать" элементы полученного массива достаточное число раз (это число задать случайным образом). Для этого номера элементов массива, значения которых будут меняться местами, задавать случайным образом.

5.2. "Модель лототрона 2". Сформировать массив размерностью "n", заполненный натуральными числами из любого произвольного диапозона. Числа в полученном массиве не должны повторяться и должны быть выработаны с помощью генератора случайных чисел. Результат работы программы проверить с помощью другого алгоритма, который тестирует результаты предыдущего алгоритма, проверяя есть или нет в сформированном массиве повторяющиеся числа.

5.3. Сформировать на основании анализа массива из "n" случайных чисел двумерный массив (или два одномерных) со следующей структурой:

- верхний ряд чисел - числа исходного массива, но записанные не более одного раза;
- нижний ряд чисел - сколько раз соответствующее число из верхнего ряда встречается в исходном массиве.

На свободные места записать нули (их нет в исходном массиве).

Например:

Исходный массив: 4,3,4,5,3,2,9,4,3,9,1,6,1,6,5,6,3

Верхний ряд: 4,3,5,2,9,1,6,0,0,0,0,0,0,0,0,0,0

Нижний ряд: 3,4,2,1,1,2,3,0,0,0,0,0,0,0,0,0,0

Рассортировать "верхний ряд" чисел по частоте с которой данное число встречается в исходном массиве. Т.е. после сортировки, для нашего примера, он будет выглядеть так:
3,4,6,5,1,2,9,0,0,0,0,0,0,0,0,0,0

Примеры решения задач.

```
{**************** Раздел описаний ****************}
uses crt; ( подключение внешнего модуля crt )
const n=10;  ( задание числа элементов массива )
var i,j,s,z,max,min,k,m:integer;   (раздел описания переменных)
```

```
  x,y:array[1..n] of integer;  (описание массива)

{****************** Раздел операторов ******************}
Begin
  window(5,5,30,10);       {создание текстового окна}
  TextBackGround(7);       {задание цвета окна}
  TextMode(1);             {задание номера текстового режима}
  TextColor(6);            {задание цвета символов}
  ClrScr;                  {перекрашивание окна в заданный цвет}
{ Создание и вывод на экран элементов массива заполненного случайными числами из
заданного диапазона}
  randomize; {установка исходного случайного числа по часам ЭВМ}
  For i:=1 to n do       {цикл "для", в котором i изменяется, как
                          номер элемента массива, от 1 до n}
    begin
     x[i]:=random(15)-5;
{заполнение i-элемента  массива случайным числом из диапазона от -5 до 10}
     write(X[I],' ');     {вывод значения i-элемента массива на экран
                           в одну строку с другими через пробел}
    end;
  writeLn;         {перевод курсора на следующую строку}
{ Альтернативный ввод элементов массива с клавиатуры }
  For i:=1 to n do
    begin
     write(' X[',i,']='); {вывод комментария}
     readLn(X[I]);              {считывание с клавиатуры значения величины
                           элемента массива}
    end;
    writeLn;
{****************** З А Д А Ч А  1.2 ******************}
{ Вычислить среднее арифметическое всех величин элементов массива.}
    s:=0;   {обнуление "счетчика"  суммы}
    For i:=1 to n do
      begin
     s:=s+x[i];   {рекурентное выражение для вычисления суммы элементов
                   массива ("счетчик" суммы)}
      end;
    writeLn (' Среднее арифметическое всех элементов массива =',s/n:5:2);

{****************** З А Д А Ч А  1.3 ******************}
{ Найти максимальное значение из всех величин элементов массива.}

    max:=x[1];   {создание "эталона" для последующих сравнений}
    For i:=2 to n do if x[i]>max then max:=x[i];
  writeLn ('Максимальное значение из всех величин элементов массива=',max);

{****************** З А Д А Ч А  1.4 ******************}
{ Найти номер элемента массива величина, которого максимальна.}
    max:=x[1];   {создание "эталона" для последующих сравнений - первый
```

```
                элемент массива)
  k:=1;  {запомнить номер элемента массива взятого в качестве эталона}
  For i:=2 to n do  {перебор всех остальных элементов массива}
  if x[i]>max then
    begin
   max:=x[i];  {занести в переменную (ячейку ОЗУ) max значение
                текущего i-го элемента массива, если он больше значения,
                которое хранится в этой переменной)
    k:=i;       {записать в переменную k номер этого элемента массива}
  end;
  writeLn ('Номер элемента массива величина, которого максимальна=',k);

{******************* З А Д А Ч А  2.1 *********************}
{Найти минимальный нечетный элемент массива }
  min:=x[1];  {создание "эталона" для последующих сравнений - первый
               элемент массива)
  For i:=2 to n do  {перебор всех остальных элементов массива}
  if (x[i]>min) and (x[i]/2<>int(x[i]/2)) then
    begin
   min:=x[i];  {занести в переменную (ячейку ОЗУ) min значение
                текущего i-го элемента массива, если он больше значения,
                которое хранится в этой переменной  и если он нечетный}
  end;
  writeLn ('Величина минимального нечетного элемента=',min);

{******************* З А Д А Ч А  2.2 *********************}
{Поменять местами минимальный элемент массива с максимальным.}
  max:=x[1];  {создание "эталонов" для последующих сравнений - первый
  min:=x[1];  элемент массива}
  k1:=1;      {запомнить номер элемента массива взятого в качестве
  k2:=1;      эталонов для максимального и минимального числа)
  For i:=2 to n do  (перебор всех остальных элементов массива)
   begin
   if x[i]>max then
     begin
       max:=x[i];    {занести в переменную max значение
                      текущего i-го элемента массива, если он больше
                      значения, которое хранится в этой переменной}
       k1:=i;  {записать в переменную k1 номер этого элемента массива}
     end;
    if x[i]<min then
     begin
       min:=x[i];          {занести в переменную min значение текущего i-го
                      элемента массива, если он меньше значения,
                      которое хранится в этой переменной}
       k2:=i;  {записать в переменную k2 номер этого элемента массива]
     end;
   end;
  x[k1]:=min; {меняем местами содержимое элемента массива величина
```

```
  x[k2]:=max;          которого имеет максимальное значение с элементом
                  массива величина которого имеет минимальное значение}
  writeLn ('Вид массива после обмена :');
  For i:=1 to n do  write(X[I],' ');

{******************** З А Д А Ч А  2.3 ********************}
{Поменять попарно равноудаленные от краев массива элементы.}

For i:=1 to n div 2 do       {сколько в массиве пар - столько раз выполняется
                        цикл, i изменяется от 1 до середины массива}
begin
  z:=x[i];                   {организация обмена содержимого элементов массива
  x[i]:=x[n-i+1];            равноудаленных от его краев через промежуточную
  x[n-i+1]:=z; переменную z}
end;
writeLn ('Вид массива после обмена равноудаленных от краев элементов:');
For i:=1 to n do  write(X[I],' ');

{******************** З А Д А Ч А  2.4 ********************}
{Поменять попарно соседние элементы массива (1 и 2, 3 и 4, и т.д.).}
For i:=1 to n div 2 do       {сколько в массиве пар - столько раз выполняется
                        цикл}
  begin
    z:=x[i*2];  {организация обмена содержимого соседних элементов
    x[i*2]:=x[n*2-1];        массива через промежуточную
    x[n*2-1]:=z;             переменную z (i*2 - четные элементы массива)}
  end;
writeLn ('Вид массива после обмена соседних элементов:');
For i:=1 to n do  write(X[I],' ');

{******************** З А Д А Ч А  3.1 ********************}
{Сдвинуть все элементы массива на k позиций вправо (k задается с клавиатуры).
  write('На сколько позиций сдвигать вправо?');
  readln(k);
  For i:=1 to k do
    For i:=1 to n-1 do       { этот цикл сдвигает массив на одну позицию влево,
                        а первый элемент становится последним}
   begin
     z:=x[i];                {организация обмена содержимого соседних
     x[i]:=x[n+1];           элементов массива через промежуточную
     x[n+1]:=z;                   переменную z (i*2 - четные элементы массива)}
 end;
writeLn ('Вид массива после сдвига на k позиций вправо:');
For i:=1 to n do
 begin
  write(X[I],' ');
 end;

{******************** З А Д А Ч А  3.2 ********************}
```

```
{Рассортировать (упорядочить) элементы массива по возрастанию методом перестановок}
for j:=1 to n-1 do              {повторяется столько раз, сколько надо
                                "поднять пузырьков"}
  for i:=1 to n-j do   {"подъем пузырька" снизу до незанятого места}
   if  x[i]>x[i+1] then
     begin
       s:=x[i];          {если "нижний"(предыдущий) элемент меньше
       x[i]:=x[i+1];  последующего он меняется с ним
       x[i+1]:=s                с помощью промежуточной ячейки s}
     end;
writeLn ('Упорядоченный массив:');
For i:=1 to n do write('X[',i,']=',X[I],' ');

{******************** З А Д А Ч А  4.1 ********************}
{Рассортировать (упорядочить) элементы массива по возрастанию методом выбора }
for j:=1 to n-1  do    {повторить n-1 раз поиск максимального элемента}
  begin
    max:=x[1];k:=1;{"образец" для поиска и его номер}
    for i:=2 to n-j+1 do     {поиск максимума от 2 до "не рассортированного"}
   if x[i]>max then
     begin
       max:=x[i];
       k:=i;
     end;
s:=x[k];          {обмен максимального
x[k]:=x[n-j+1];       с крайним "не рассортированным"}
x[n-j+1]:=s
end;
writeLn ('Упорядоченный массив:');
For i:=1 to n do write(X[I],' ');

repeat until keypressed;
end.
```

IV. Задачи по теме «Двумерные массивы»

На итоговой контрольной необходимо решить 3 задачи. На 3 - одну задачу (стандартная задача, комментарии и блок схема к ней), на 4 - любую пару задач, на 5 – все три. Для любой из задач должна быть изображена блок схема алгоритма и к каждой строке должны быть написаны подробные комментарии. Также должна быть предъявлена дискета с задачами отлаженными в классе.

На зачете или экзамене, кроме решения задачи, необходимо ответить письменно или устно на контрольные вопросы указанные в билете, а также объяснить подробно один из стандартных алгоритмов.

P. S. Те учащиеся для которых уровень предлагаемых задач слишком низок могут решить три задачи повышенной сложности, а затем задачи из районных олимпиад по программированию.

Общее условие для всех задач:

Создать в памяти ЭВМ двумерный массив из n-целых чисел (n - константа) с помощью генератора случайных чисел, по формуле или задавая значения элементов массива с клавиатуры. После запуска созданной Вами программы на экран должны быть выведены исходные значения всех элементов массива и результат его обработки.

БЛОК 1

Стандартные алгоритмы.

1.1. Заполнить квадратную матрицу таким образом чтобы на главной диагонали находились единицы, а на остальных местах нули.

Самостоятельная работа №1 на ЭВМ.

1.2. Заполнить квадратную матрицу знаком "*" и пробелами таким образом чтобы она соответствовала флагу Великобритании (учитывая цвет).

Домашнее задание №1.

1.3. Вычислить среднее арифметическое всех четных величин матрицы.

1.4. В матрице найти среднеарифметическое для элементов каждого столбца.

1.5. В матрице найти среднеарифметическое для элементов каждой строки.

1.6. В матрице найти среднеарифметическое для элементов главной диагонали.

1.7. В матрице найти среднеарифметическое для элементов расположенных выше главной диагонали.

1.8. В матрице найти среднеарифметическое значение всех элементов кратных трем.

1.9. Вычислить разность между количеством элементов массива величина которых - четное целое число и количеством элементов массива величина которых – нечетное целое число.

1.10. Вычислить разность между средним арифметическим для элементов массива величина которых - четное целое числах и средним арифметическим для элементов массива величина которых не нечетное целое число.

1.11. Найти в матрице местоположение максимального элемента.

1.12. Найти в матрице местоположение минимального элемента.

1.13. В матрице найти местоположение первого элемента кратного трем.

1.14. В матрице найти местоположение максимального элемента кратного трем.

1.15. Вычислить разность между максимальным и минимальным элементами матрицы.

1.16. Определить число строк матрицы в которых есть число пять.

1.17. Вычислить для матрицы среднеарифметическое значение для элементов каждой строки.

1.18. Вычислить для матрицы число нечетных чисел в каждом столбце.

1.19. Поменять местами минимальное и максимальное число в матрице.

БЛОК 2.

Стандартные алгоритмы.

2.1. Поменять местами попарно в каждом столбце соседние элементы массива (например в 3 столбце элементы 1,3 и 2,3; 3,3 и 4,3; 5,3 и 6,3; и т.д.).

2.2. В матрице найти столбец, в котором сумма элементов минимальна.

2.3. Рассортировать элементы в строке, номер которой задается с клавиатуры.

Самостоятельная работа №2.

2.4. (четные ЭВМ). В матрице найти строку, в котором сумма элементов максимальна и рассортировать их в ней.

2.5. (нечетные ЭВМ). В матрице найти строку, в котором сумма элементов минимальна и поменять местами в ней элементы равноудаленные от ее краев.

Домашнее задание №2.

2.6. В матрице найти максимальный элемент в столбце, в котором находится минимальный элемент.

2.7. В матрице найти строку, в которой сумма элементов минимальна. Какой элемент в этой стро-

ке максимальный?

2.8. В матрице найти местоположение максимального элемента кратного трем и в столбце где он находится отыскать самый нижний нечетный элемент.

2.9. Поменять местами элементы столбца в котором находится максимальный элемент квадратной матрицы с элементами строки в которой находится в которой находится минимальный элемент.

2.10. В матрице найти максимальный элемент в строке, в котором находится минимальный элемент.

2.11. В матрице найти строку, в которой произведение элементов минимально. Какой четный элемент в этой строке максимальный?

2.12. В матрице найти местоположение максимального элемента кратного двум и в строке где он находится отыскать минимальный нечетный элемент.

2.13. Поменять местами элементы строки в которой находится максимальный четный элемент квадратной матрицы элементами строки в которой находится в которой находится минимальный элемент.

2.14. В матрице найти минимальный элемент в столбце, в котором находится максимальный элемент.

2.15. В матрице найти строку, в которой количество четных элементов минимальна. Какой элемент в этой строке максимальный?

2.16. В матрице найти местоположение максимального нечетного элемента и в столбце где он находится определить число нечетных элементов.

2.17. Поменять местами элементы столбца в котором находится минимальный элемент квадратной матрицы с элементами последней строки.

2.18. В матрице найти максимальный элемент в столбце, в котором находится больше всего четных чисел.

2.19. В матрице найти строку, в которой произведение элементов минимально. Какой элемент в этой строке минимальный?

2.20. В матрице найти местоположение максимального элемента кратного трем и в столбце где он находится отыскать самый нижний нечетный элемент.

2.21. Поменять местами элементы столбца в котором находится максимальный элемент квадратной матрицы с элементами первой строки.

БЛОК №3.

Стандартные алгоритмы.

3.1. В строке в которой находится минимальный элемент массива найти максимальный элемент и поменять его с этим элементом.

3.2. Поменять местами столбцы с максимальной и минимальной суммой элементов.

3.3. Рассортировать все строки массива по убыванию.

Самостоятельная работа №3

3.4 (1, 5, 9 ЭВМ). Поменять местами максимальный элемент того столбца матрицы, в которой находится минимальный элемент всей матрицы с минимальным элементом того столбца, в котором находится максимальный элемент всей матрицы.

3.5. (2, 6, 10 ЭВМ). Рассортировать по убыванию столбец в котором находится максимальный элемент всей матрицы.

3.6. (3, 7 ЭВМ). Поменять местами строки с максимальным и минимальным произведением четных элементов.

3.7. (4, 8 ЭВМ). Рассортировать по возрастанию строку с минимальным произведением четных элементов.

БЛОК 4.

Задачи повышенной сложности.

4.1. ЗАДАЧА ТЕХНИЧЕСКОГО ЗРЕНИЯ. На конвейере двигается много разных деталей. По "эталону" конкретной детали, робот должен ее распознать, даже если она сдвинута, повернута или перевернута.

Заданы две матрицы заполненные нулями и единицами. Единицы в каждой матрице сгруппированы в многоугольник. Одна матрица - эталон. Вторая - изображение детали на конвейере. Составить программу которая определяет, что в обоих матрицах один и тот же многоугольник, но:

- сдвинутый параллельно относительно одной или двух сторон матрицы;
- или повернутый на 90, 180 или 270 градусов;
- или зеркально отраженный относительно одной из своих сторон;
- или все это вместе.

Иначе программа должна выдать ответ : "Это не та деталь!".

4.2. Рассортировать весь массив "змейкой" - за последним элементом каждого столбца должен идти отсортированный первый элемент следующего столбца.

4.3. Составить программы шифратор и дешифратор действующие по принципу трафарета, который накладывается на двумерный массив. "Закрытые" позиции заполняются с помощью генератора случайных чисел. Преимущество метода - им можно пользоваться и без компьютера, делая на нем лишь трафареты и зашифровывая текст.

Примеры решения задач

```
{ Общая часть }

 Uses Crt;                    {подключение внешнего модуля crt}
{-------------------------------------------------------------------------}
  Const n=5; m=5;                    {задание числа элементов массива}
  Var
    i, j, f, c, Max, Min, k, l, g, f: integer;   { задание целых переменных }
    X1:Array[1..N,1..M] of integer; {описание целочисленного двумерного массива }
    X2: Array [1..N, 1..M] of char;  {задание двумерного массива символьных переменных}
{———————— СОЗДАНИЕ МАССИВА————————————}
Begin
  ClrScr;                    {очистка, закраска окна заднего плана}
  Randomize;
  for i:=1 to n do           {изменение номера столбца}
    for j:=1 to m do                 {изменение номера строки}
      begin
        x1[i,j]:=random(100);        {задание случайного элемента под номером i, j}
                                     {в пределах 100}
        GoToXY(i*3+3,j+2);           {перевод курсора на позицию i*3+3, j+2}
        Write(x1[i,j]);              {вывод элемента двумерного массива на экран}
      end;

{1.1. Заполнить квадратную матрицу таким образом чтобы на главной диагонали находились единицы, а на остальных местах нули.}
For i:=1 to n do
  For j:=1 to m do
    begin
      if  i=j  then  x[i,j]:=1  else x[i,j]:=0;
      goToXY(i*2,j+12);
      write(X[i,j]);
    end;
{2.1. Поменять  местами попарно в каждом столбце соседние элементы массива
(например, в 3 столбце элементы 1,3 и 2,3; 3,3 и 4,3; 5,3 и 6,3; и т.д.).}
For j:=1 to m do
```

```
  For i:=1 to n div 2 do
    begin
      k:=x[i*2-1,j];
      x[i*2-1,j]:=x[i*2,j];
      x[i*2,j]:=k;
    end;
For i:=1 to n do
  For j:=1 to m do
    begin
      GoToXY(i*2+40,j);
      write (X[i,j]);
    end;

   {2.2. В матрице найти столбец, в котором сумма элементов минимальна.}
j:=1; min:=0;
for i:=1 to n do
  min:=min+x1[i,j];
  k:=1;
  for j:=2 to m do
    begin
      f:=0;
      for i:=1 to n do
        f:=f+x1[i,j];
        if f<min then
        begin
          min:=f;
          k:=j;
        end;
    end;
for i:=1 to n do
  for j:=1 to m do
    begin
      if J=k then TextColor(9)
        else  TextColor(15);
        GoToXY(i*3+3,j+2);
        Write(x1[i,j]);
    end;

{2.3. Рассортировать элементы в строке, номер которой задается с клавиатуры}
write(' В какой строке будем сортировать?');
readLn(k);
for a:=1 to n-1
  for i:=1 to m-a do
    IF x[i,k]>x[i+1,k] then
      begin
        c:=x[i,k];
        x[i,k]:=x[i+1,k];
        x[i+1,k]:=c;
      end;
```

```
for i:=1 to n do
  for j:=1 to m do
    begin
      GoToXY(i*3+3,j+2);
      Write(x1[i,j]);
   end;
```

{3.1. В строке в которой находится минимальный элемент массива найти максимальный элемент и поменять его с этим элементом.}

```
For i:=1 to n do
  For j:=1 to m do
    begin
      goToXY (i*2+20,j+15);
      write (X[i,j]);
    end;
  writeLn;
  min:=x[1,1];
  k:=1; c:=1; l:=1;
For i:=1 to n do
  For j:=1 to m do
    if x[i,j]<min then
      begin
        min:=x[i,j];
        k:=j; c:=i;
      end;
  max:=x[1,k];
  For i:=2 to n do
    if x[i,k]>max then
      begin
        max:=x[i,k];
        l:=i;
      end;
  x[l,k]:=min;
  x[c,k]:=max;
For i:=1 to n do
  For j:=1 to m do
    begin
     goToXY(i*2+40,j+15);
     write(X[i,j]);
  end;
writeLn;
```

{3.2. Поменять местами столбцы с максимальной и минимальной суммой элементов}

```
j:=1; max:=0; min:=0
for i:=1 to n do
Begin
  max:=max+x1[i,j];
  min:=min+x1[i,j];
End;
```

```
k:=1;
L:=1;
for j:=2 to m do
  begin
    f:=0;
    g:=0;
    for i:=1 to n do
      f:=f+x1[i,j];
      if f>max then
        begin
          max:=f;
          k:=j;
        end;
      if g<min then
        begin
          min:=g;
          L:=j;
        end;
   end;
For i:=1 to n do
  Begin
    c:=x1[i,l];
    x1[i,l]:=x1[i,k];
    x1[i,k]:=c;
  End;

for i:=1 to n do
  for j:=1 to m do
    begin
      GoToXY(i*3+3,j+2);
      Write(x1[i,j]);
    end;
readln;

{3.3. Рассортировать все строки массива по убыванию.}
For i:=1 to n do
  begin
  For jj:=1 to m-1 do
    begin
      min:=x[i,1];
      c:=1;
      For j:=1 to m-jj+1 do
       if x[i,j]<min then
       begin
          min:=x[i,j];
          c:=j;
       end;
       x[i,c]:=x[i,m-jj+1];
       x[i,m-jj+1]:=min;
```

```
    end;
  end;

For i:=1 to n do
  For j:=1 to m do
    begin
      goToXY(i*2+40,j+20);
      write(X[i,j]);
    end;
writeLn;
Readln;
End/
```

V. Задачи по теме «Строковые величины»

БЛОК № 1.

Стандартные алгоритмы,

в которых не используются символьные функции и процедуры, кроме функции определяющей длину строки.

1.1. Преобразовать слово "АПЕЛЬСИН" в слово "СПАНИЕЛЬ".
1.2. Определить сколько в строке гласных букв.
1.3. Определить сколько в строке слов.
1.4. Вывести на экран заданную строку, причем каждое новое слово выводить с новой строки.
1.5. Организовать вывод на экран строки в стиле "падающие буквы".

Самостоятельная работа №1 на ЭВМ.

1.6. (четные ЭВМ). Определить является ли слово веденное с клавиатуры полиндромом (пример полиндрома: "топот").
Определить какой в нем процент шипящих звуков. Организовать вывод каждой буквы этого слова на экран через пробел и разного цвета.
Организовать вывод на экран этого слова по одной букве, причем буква должна "пробежать" по экрану справа налево до своего места.
1.7. (нечетные ЭВМ). Определить является ли слово веденное с клавиатуры одним из дней недели.
Определить какой в нем процент глухих согласных. Организовать вывод этого слова на экран сверху вниз (по вертикале), каждая буква должна быть разного цвета.
Организовать вывод на экран этого слова по одной букве, причем буква должна "пробежать" по экрану слева направо до своего места.

Домашнее задание №1.

1.8. Преобразовать слово "АБИТУРИЕНТ" в "БУНТ" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.9. Преобразовать слово "ГЛУБОКОУВАЖАЕМЫЙ" в "БОМЖ" и расположить четыре таких слова на экране по четырем сторонам ромба.
1.10. Преобразовать слово "ЛАПОТЬ" в "ТОПОЛЬ" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.11. Преобразовать слово "МЕТАМОРФОЗА" в "МЕТАФОРА" и расположить четыре таких слова на экране по четырем сторонам ромба.
1.12. Преобразовать слово "МОРЕПЛАВАТЕЛЬ" в "ВАЛЕРА" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.13. Преобразовать слово "ОРДИНАРЕЦ" в "ОРДА" и расположить четыре таких слова на экране по четырем сторонам ромба.

1.14. Преобразовать слово "ПУСТЯЧНЫЙ" в "ПУТЧ" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.15. Преобразовать слово "СПАРТАК" в "КАРТА" и расположить четыре таких слова на экране по четырем сторонам ромба.
1.16. Преобразовать слово "ЯЗЫКОЗНАНИЕ" в "КОЗА" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.17. Преобразовать слово "ГЛУБОКОУВАЖАЕМЫЙ" в "БОМЖ" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.18. Преобразовать слово "ЛАПОТЬ" в "ТОПОЛЬ" и расположить четыре таких слова на экране по четырем сторонам ромба.
1.19. Преобразовать слово "МЕТАМОРФОЗА" в "МЕТАФОРА" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.20. Преобразовать слово "МОРЕПЛАВАТЕЛЬ" в "ВАЛЕРА" и расположить четыре таких слова на экране по четырем сторонам ромба.
1.21. Преобразовать слово "ПУСТЯЧНЫЙ" в "ПУТЧ" и расположить четыре таких слова на экране по четырем сторонам ромба.
1.22. Преобразовать слово "СПАРТАК" в "КАРТА" и расположить четыре таких слова на экране по четырем сторонам квадрата.
1.23. Преобразовать слово "ЯЗЫКОЗНАНИЕ" в "КОЗА" и расположить четыре таких слова на экране по четырем сторонам ромба.

БЛОК № 2.
Стандартные алгоритмы,
использующие определение позиции подстроки, удаление, копирование и вставку части строки.
2.1. Определить сколько раз заданное слово встречается в строке.
2.2. Напечатать заданную строку, удалив из нее лишние пробелы (оставить один между словами).
2.3. Во всей строке заменить одно заданное слово на другое.

Самостоятельная работа №2 на ЭВМ.
2.4. Вывести на экран из заданной строки ключевое слово, которое вводится с клавиатуры и следующее за ним слово. Если ключевых слов в строке несколько, то выводить каждое новое с следующим за ним с новой строки новым цветом. Вычислить каков процент ключевых слов в строке.

Домашнее задание №2.
2.5. В заданной строке поменять местами пары символов расположенные на четных и нечетных местах.
2.6. В заданной строке поменять местами символы равноудаленные от краев строки.
2.7. Определить в каком месте строки последний раз встречается буква 'а'.
2.8. Напечатать слово 'False', если в строке буква 'а' встречается чаще чем буква "о".
2.9. Из заданной строки удалить все пробелы.
2.10. Определить сколько слов в строке начинаются на заданную букву.
2.11. Удвоить каждую букву в заданной строке.
2.12. Увеличить во всей строке интервал между словами с 1 пробела на 2.
2.13. В строке сдвинуть все символы на заданное число знакомест.
2.14. В заданной строке поменять местами символы равноудаленные от краев строки.
2.15. Определить в каком месте строки последний раз встречается буква 'а' в начале слова.
2.16. Напечатать какая буква встречается в строке максимальное число раз.
2.17. Из строки удалить все слова которые начинаются на заданную букву.
2.18. Определить сколько слов в строке кончается на заданную букву.
2.11. Удвоить в строке каждую букву в заданном слове.
2.12. Добавить в заданном слове строки пробелы между буквами.

БЛОК № 3.
Стандартные алгоритмы,
использующие определение кода символа и символа по его коду.

3.1. В заданной строке все строчные буквы заменить на прописные.
3.2. Преобразовать строку состоящую из арабских цифр в число.
Самостоятельная работа №3 на ЭВМ.
3.3 В заданной строке все строчные буквы в начале каждого слова заменить на прописные.
Зашифровать текст сдвинув коды всех символов. Вывести его на экран.
Расшифровать текст, зашифрованный при выполнении предыдущего пункта задания.

Задачи повышенной сложности.
4.1. Организовать вывод на экран текста любой длины в виде бесконечной "бегущей строки".
4.2. Зашифровать текст с помощью кодировочной таблицы.
4.3. Расшифровать текст зашифрованный при выполнении предыдущей задачи, используя кодировочную таблицу.
4.4. Определить является ли данная строка палиндромом (пример полиндрома:
"А роза упала на лапу Азора".
4.5. Проверить, правильно ли в заданной строке расставлены круглые скобки.
4.6. Организовать вывод текста на экран как массива строк. Организовать поиск в этом массиве ключевого слова, причем курсор должен перемежаться на экране сначала в начало первого ключевого слова, затем следующего и т.д.

Примеры решения задач.

```
uses crt;
var
   j,i:byte;
   k:integer;
   a:char;
   s,z,d:string;
   flag:boolean;

Begin                    {Общая часть}
Clrscr;
Writeln ('Введите строку символов:');
Readln (s);

{1.2. Определить сколько в строке гласных букв.}
WriteLn(s);
j:=0;
for i:=1 to length(s)  do
  if (s[i]='а') or (s[i]='о') or (s[i]='е')
    or (s[i]='у') or (s[i]='я') or (s[i]='э')
    or (s[i]='ю')  or (s[i]='ы') or (s[i]='и')
  then j:=j+1;
writeLn ('Ответ 1.2: ', j);
Readln;

{1.3. Определить сколько в строке слов.}
WriteLn (s);
j:=0; s:=' '+s;
for i:=1 to length(s)-1  do
  if (s[i]=' ') and (s[i+1]<>' ') then j:=j+1;
writeLn ('Ответ 1.3: ', j);
```

```
repeat until keypressed;

{1.4. Вывести на экран заданную строку, причем каждое новое слово выводить с новой строки.}
WriteLn(s);
z:=''; s:=s+' *';
gotoXY (1,10);
for i:=1 to length(s)-1  do
  begin
    z:=z+s[i];
    if (s[i]=' ') and (s[i+1]<>' ') then
      begin
        writeLn;
        write (z);
        z:='';
      end;
    end;

{1.5. Организовать вывод на экран строки в стиле "падающие буквы".}
WriteLn(s);
Randomize;
for i:=1 to length(s)  do
  begin
    for j:=1 to 10  do
      begin
       TextColor(j); gotoXY(i,j);
       writeLn(s[i]);
       delay(2);
       TextColor(0); gotoXY(i,j); writeLn(' ');
       end;
    TextColor (random(7)+9); gotoXY(i,j); writeLn(s[i]);
  end;

{2.1. Определить сколько раз заданное слово встречается в строке.}
s:='роза упала роза на лапу роза азора';
WriteLn(s);
s:=' '+s+' ';i:=0;
z:=' роза ';
if pos(z,s)<>0 then
  repeat
    delete(s,pos(z,s),length(z));
    i:=i+1;
  until pos (z,s)=0;
if i<>0 then writeLn ('Ответ 2.1:',i)
        else writeLn ('Ответ 2.1: Нет такого слова!');

{2.2. Напечатать заданную строку, удалив из нее лишние пробелы (оставить один между словами).}
s:='а   роза   упала   на   лапу   азора';
```

```
WriteLn(s);
z:=' ';
for i:=1 to length(s)-1  do
  if not ((s[i]='  ') and (s[i+1]='  ')) then z:=z+s[i];
z:=z+s[length(s)];
writeLn ('Ответ 2.2:', z);

{2.3. Во всей строке заменить одно заданное слово на другое.}
s:='роза упала роза на лапу роза азора';
d:='мимоза';
z:='роза';
WriteLn (s);
i:=0;
if pos (z, s) <> 0  then
  begin
    repeat
      j:= pos (z, s);
      delete (s, j, length(z));
      insert (d, s, j);
    until pos (z, s)=0;
    writeLn ('Ответ 2.3 :', s);
  end
else writeLn ('Ответ 2.3: Нет слова ', z);

{3.1. В заданной строке все строчные буквы заменить на прописные.
      Коды русских букв:          от "А" до "П" - от 128 до 143
                                  от "Р" до "Я" - от 144 до 159
                                  от "а" до "п" - от 160 до 175
                                  от "р" до "я" - от 224 до 239
                                  "ё" - 241, "Ё" - 240 }

s:='абвгдежзиклмнопрстфхцчшщэюяьыё';
WriteLn(s);
z:='';
for i:=1 to length(s)  do
  begin
    if (ord(s[i])<176) and (ord(s[i])>159) then  s[i]:=chr(ord(s[i])-32);
    if (ord(s[i])<240) and (ord(s[i])>223) then  s[i]:=chr(ord(s[i])-80);
    if ord(s[i])=241 then  s[i]:=chr(240);
  end;
writeLn ('Ответ 3.1 :', s);

{3.2. Преобразовать строку состоящую из арабских цифр в число.}
{      (1000, 900, 500, 400, 100, 90,  50,  40,  10,   9,  5,   4,   1);
       ( 'M',  'CM', 'D',  'CD', 'C', 'XC', 'L', 'XL', 'X', 'IX', 'V', 'IV', 'I');}
while s<>'' do
begin
  if s[1]='M' then   begin n:=n+1000; delete(s,1,1); end;
  if s[1]='C' then
```

```
    if s[1]+s[2]='CM' then   begin n:=n+900; delete(s,1,2); end
      else  if s[1]+s[2]='CD' then begin n:=n+400; delete(s,1,2); end
          else begin n:=n+100; delete(s,1,1); end;

if s[1]='D' then   begin n:=n+500; delete (s, 1, 1); end;
if s[1]='X' then
     if s[1]+s[2]='XC' then      begin n:=n+90; delete (s, 1, 2); end
       else if s[1]+s[2]='XL' then begin n:=n+40; delete (s, 1, 2); end
            else begin n:=n+10; delete (s, 1, 1); end;
if s[1]='L' then   begin n:=n+50; delete (s, 1, 1); end;
if s[1]='I' then
    if s[1]+s[2]='IX' then       begin n:=n+9; delete (s, 1, 2); end
      else if s[1]+s[2]='IV' then  begin n:=n+4; delete (s, 1, 2); end
           else begin n:=n+1; delete (s, 1, 1); end;
if s[1]='V' then   begin n:=n+5; delete (s, 1, 1); end;

Repeat Until Keypressed
End.
```

VI. Задачи по теме «Множества»

Требования к итоговой аттестации.

На итоговой контрольной работе необходимо решить 3 задачи. На «3» - №1, на «4» - любую пару задач, на «5» - все три. Для любой из задач должна быть изображена блок схема алгоритма и к каждой строке должны быть написаны подробные комментарии. На зачете или экзамене, кроме решения задачи, необходимо ответить письменно или устно на контрольные вопросы указанные в билете, а также объяснить подробно один из стандартных алгоритмов.

БЛОК 1. «Простейшие стандартные задачи»

Решить в тетради для домашних заданий:

1.1. Type day=(pn,vt,sr,th,pt,sb,vs); - дни недели. Описать множественный тип, включающий в себя множества из названий любых дней недели.

1.2. Type day=(pn,vt,sr,th,pt,sb,vs); - дни недели. Описать множественный тип, включающий в себя множества из названий рабочих дней недели.

1.3. Какие из следующих конструкций являются множествами (в смысле языка Паскаль), а какие нет и почему? а) [9,6,3,0]; б) [2..3,5,7]; в) [1..15,4..18]; г) [9/3, 4, 5]

1.4. Какие из следующих конструкций являются множествами (в смысле языка Паскаль), а какие нет и почему: а) [0..0]; б) ['*','*']; в) [true..false];

1.5. Какие из следующих конструкций являются множествами (в смысле языка Паскаль), а какие нет и почему: а) [2,sqrt(9)]; б) ['=','>=','>']; в) [[],[5]];

1.6. Var p:set of 0..9;
i,j : integer;

Если i=3 и j=5 то какое значение получит переменная P при выполнении следующего оператора присваивания: а) p:=[i+3, j div 2, j..sqr(i)-3]; б)p:=[2*i..j];в) p:=[i,j,2*i,2*j];

1.7. Var s:set of char; Переменной S присвоить: а) Пустое множество; б) множество из (a,e,i,o,u); в) множество из всех цифр.

1.8. Вычислить значения отношений: а) [2]<>[2,2,2]; б)['a','b']=['b','a']; в)[4,5,6]=[4..6];

1.9. Вычислить значения отношений: а) ['c','b']=['c'..'b']; б) [2,3,5,7]<=[1..9]; в) [3,6..8]<=[2..7,9];

1.10. Вычислить значения отношений: а) []<=['0'..'9']; б) 'q' in ['a'..'z']; в) [2]<=[1..3];

г) 55=[55];

1.11. Type mount=1..12; Описать функцию NumDay(m), определяющую количество дней в месяце m (невисокосного года).

1.12. Type M=set of 0..99; Описать функцию Card(A), подсчитывающую количество элементов в множестве A типа M (например, Card([5,8,23]).

1.13. Type letter=set of 'a'..'z'; Описать процедуру Print(A), печатающую в алфавитном порядке все элементы множества A. имеющего тип letter.

1.14. Type Day=(pn,vt,sr,th,pt,sb,vs); - дни недели. WorkDay=pn..pt; var wd : day; t : boolean; Требуется переменной t присвоить значение true, если wd - рабочий день, и значение false - иначе. Какими из следующих операторов правильно решается эта задача: а) t:=wd in WorkDay; б) t:=wd=WorkDay; в) t:=wd in [WorkDay];

1.15. Условие задачи такое же, как в задаче 1.14: а) t:=wd in [pn..pt]; б) t:=[wd]<=[pn..pt]; в) t:=[wd]=[pn..pt].

1.16. Вычислить значения выражений:

а) [2..13]*[3,13..60]+[4..10]-[5..15]*[6];

б) [2..10]-[4,6]-[2..12]*[8..15];

в) (['0'..'7']+['2'..'9'])*(['a']+['z']).

Общее условие для всех задач из блоков 2,3,4:

Создать в памяти ЭВМ два множества (X и Y), введя их элементы с клавиатуры, или используя случайные числа или с помощью конструктора. Диапазон значений величин элементов множеств и их тип задается учителем. На экран должны быть выведены исходные значения всех элементов множеств и результаты их обработки после выполнения программы.

БЛОК №2.

Стандартные алгоритмы

2.1. Вывести на экран пересечение, объединение и разность двух множеств.

2.2. Определить количество элементов во множестве.

2.3. Вывести на экран четные элементы числового множества.

2.4. Подсчитать количество нечетных элементов во множестве.

Самостоятельная работа №2.

2.5. Определить количество элементов во множестве, которое является пересечением двух других множеств (ЭВМ № 1, 4, 7, 10).

2.6. Определить количество элементов во множестве, которое является объединением двух других множеств (ЭВМ № 2, 5, 8).

2.7. Определить количество элементов во множестве, которое является разностью двух других множеств (ЭВМ № 3, 6, 9).

Домашнее задание № 2.

2.8. Вывести на экран четные элементы числового множества, которое является пересечением двух других множеств.

2.9. Вывести на экран нечетные элементы числового множества, которое является объединением двух других множеств.

2.10. Вывести на экран кратные трем элементы числового множества, которое является разностью двух других множеств.

2.11. Вывести на экран нечетные элементы числового множества, которое является пересечением двух других множеств.

2.12. Вывести на экран кратные пяти элементы числового множества, которое является объединением двух других множеств.

2.13. Вывести на экран нечетные элементы числового множества, которое является разностью двух других множеств.

2.14. Подсчитать количество нечетных элементов во множестве, которое является пересечением двух других множеств.

2.15. Подсчитать количество четных элементов во множестве, которое является объединением двух других множеств.

2.16. Подсчитать количество кратные трем элементов числового множества, которое является раз-

ностью двух других множеств.

2.17. Подсчитать количество четных элементов во множестве, которое является пересечением двух других множеств.

2.18. Определить количество кратных пяти элементов числового множества, которое является объединением двух других множеств.

2.19. Вывести на экран количество нечетных элементов числового множества, которое является разностью двух других множеств.

БЛОК №3.

Стандартные алгоритмы

3.1. Найти сумму всех элементов множества.

3.2. Найти среднее арифметическое всех элементов множества.

3.3. Удвоить значения всех элементов множества.

3.4. Найти максимальный элемент в числовом множестве и вывести на экран

Самостоятельная работа №3.

3.5. Найти сумму всех нечетных элементов множества (ЭВМ № 1, 6).

3.6. Найти среднее арифметическое всех четных элементов множества (ЭВМ № 2, 7).

3.7. Найти первый по алфавиту элемент в символьном множестве и вывести его на экран. (ЭВМ № 3, 8).

3.8. Создать числовое множество, у которого значения элементов равны разности элементов исходного множества с его минимальным элементом. (ЭВМ № 4, 9)

3.9. Создать числовое множество, у которого значения элементов равны сумме элементов исходного множества с его максимальным элементом. (ЭВМ № 5, 10)

Домашнее задание № 3.

3.10. Найти сумму четных элементов множества, которое является пересечением 2-х других множеств.

3.11. Найти сумму нечетных элементов множества, которое является объединением 2-х других множеств.

3.12. Найти сумму кратных трем элементов множества, которое является разностью 2-х других множеств.

3.13. Найти среднее арифметическое нечетных элементов множества, которое является пересечением 2-х других множеств.

3.14. Найти среднее арифметическое четных элементов множества, которое является объединением 2-х других множеств.

3.15. Найти среднее арифметическое кратных пяти элементов множества, которое является разностью 2-х других множеств.

3.16. Найти максимальный кратный трем элемент в числовом множестве.

3.17. Найти минимальный кратный пяти элемент в числовом множестве.

3.18. Найти максимальный четный элемент в числовом множестве, которое является пересечением двух других множеств.

3.19. Найти минимальный нечетный элемент в числовом множестве, которое является объединением двух других множеств.

3.20. Найти минимальный четный элемент в числовом множестве, которое является разностью двух других множеств.

3.21. Уменьшить значение всех элементов множества на значение его минимального четного элемента.

3.22. Уменьшить значение всех элементов множества на значение его максимального нечетного элемента.

3.23. Вывести на экран те элементы множества, которые меньше среднего арифметического значений его минимального и максимального элементов.

3.24. Определить, сколько элементов множества больше, чем среднее арифметическое значений его максимального и минимального элементов.

Примеры решения задач.

```
Program Mnogestva;
Uses crt;
Var X, Y, Z: set of byte;
    Xch, Ych, Zch: set of Char;
    i, j, k, Max, min, s: integer;
    ch: char;

Begin
{Ввод элементов множества X с клавиатуры}
  i:=0;
  repeat
    i:=i+1;
    write (i,'-й элемент множества X: ');
    Readln (k);
    writeln ('Любая клавиша - продолжить, Esc - хватит');
    X:=X+[k];
    ch:=readkey;
  until ch=#27; {#27 - код клавиши Esc}

{Ввод элементов множества Y с клавиатуры}
  i:=0;
  repeat
    i:=i+1;
    write (i,'-й элемент множества Y: ');
    Readln (k);
    writeln ('Любая клавиша - продолжить, Esc - хватит');
    Y:=Y+[k];
    ch:=readkey; {#27 - код клавиши Esc}
  until ch=#27;

{Вывод элементов множества X на экран}
  Write ('X=[');
  For i:=0 to 255 do
    If i in X then write (i,',');
  Writeln (#8,']'); {Здесь #8 - символ клавиши BackSpace}

{Вывод элементов множества Y на экран}
  Write ('Y=[');
  For i:=0 to 255 do
    If i in Y then write (i,',');
  Writeln (#8,']'); {Здесь #8 - символ клавиши BackSpace}

{2.1. Вывести на экран пересечение, объединение и разность двух множеств}
  Writeln ('Пересечение X и Y:');
  Write ('X*Y=[');
  For i:=0 to 255 do
    If i in X*Y then write (i,',');
  Writeln (#8,']');          {Здесь #8 - символ клавиши BackSpace}
```

```
 Writeln ('Объединение X и Y:');
 Write ('X+Y=[');
 For i:=0 to 255 do
   If i in X+Y then write (i,',');
 Writeln (#8,']');          {Здесь #8 - символ клавиши BackSpace}

 Writeln ('Разность X и Y:');
 Write ('X-Y=[');
 For i:=0 to 255 do
   If i in X-Y then write (i,',');
 Writeln (#8,']');  {Здесь #8 - символ клавиши BackSpace}

{2.2. Определить количество элементов во множестве}
k:=0;
for i:=0 to 255 do
   If i in X then k:=k+1;
Writeln ('Количество элементов во множестве X:',k);

{2.3. Вывести на экран четные элементы множества}
Write ('Четные элементы множества X=[');
for i:=0 to 255 do
   If (i in X) and (i/2=int(i/2)) then write (i,',');
   Writeln (#8,']');

{2.4. Определить количество нечетных элементов во множестве}
k:=0;
for i:=0 to 255 do
   If (i in X) and (i/2<>int(i/2)) then k:=k+1;
Writeln ('Количество нечетных элементов во множестве X:',k);

{3.1. Найти сумму всех элементов множества}
s:=0;
for i:=0 to 255 do
   If i in X then s:=s+i;
Writeln ('Сумма элементов множества X:',s);

{3.2. Найти среднее арифметическое всех элементов множества}
s:=0; k:=0;
for i:=0 to 255 do
   If i in X then Begin s:=s+i; k:=k+1; End;
Writeln ('Среднее арифметическое значение элементов множества X:',s/k:6:2);

{3.3. Удвоить значения всех элементов множества}
for i:=0 to 255 do
  If i in X then Z:=Z+[i*2];
  X:=Z;
{Вывод элементов множества X с клавиатуры}
  Write ('Удвоенные элементы множества X=[');
  For i:=0 to 255 do
```

```
  If i in X then write (i,',');
 Writeln(#8,']'); {Здесь #8 - символ клавиши BackSpace}

{3.4. Найти максимальный элемент в числовом множестве}
for i:=0 to 255 do
  If i in X then k:=i;
Writeln ('Максимальный элемент множества X:',k);

repeat until keypressed;
End.
```

Список использованной литературы.

1. Вирт Н., Йенсен К. Паскаль: Руководство для пользователя. –М.: Компьютер, 1993.-256 с.

2. Шаньгин В.Ф., Поддубная Л.М. Программирование на языке Паскаль. М.: Высшая школа. 1991.-142 с.

3. Фаронов В.В. Программирование на персональных ЭВМ в среде Турбо-Паскаль. -М.: Изд. МГТУ, 1991.– 580 с.

4. Вальвачев А.Н. Графическое программирование на языке Паскаль.: Справ. Пособие. –Мн.: Высш. шк., 1992.– 143 с.

5. Бородич Ю.С. и др. Паскаль для персональных компьютеров: Справ. пособие – Выш. шк.: БФ ГИТМП «НИКА», 1991.-365 с.

6. Фаронов В.В. Турбо Паскаль (в 3-х книгах). Книга 1. Основы Турбо Паскаля. –М.: Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992.– 304 с.

7. Емелина Е.И. Основы программирования на языке Паскаль.– М. Финансы и статистика, 1997.– 208 с.