МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Колледж электроники и бизнеса ОГУ

Л.А.ДЕЛЬ, А.А.ГУРЬЯНОВ

ПРАКТИКА ДЛЯ ПОЛУЧЕНИЯ ПЕРВИЧНЫХ ПРОФЕССИОНАЛЬНЫХ НАВЫКОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
часть 1

Рекомендовано к изданию Редакционно-издательским советом
государственного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»

Оренбург 2008

УДК 681.3.06 (075.32)
ББК 22.18я73
Д-29

Рецензент
Заместитель директора по научно-методической работе С.А.Кузюшин.

Дель Л.А.
Д-29 **Практике для получения первичных профессиональных навыков: методические указания /Л.А.Дель, А.А.Гурьянов – Оренбург: ГОУ ОГУ, 2008. - 74 с.**

Методические указания предназначены для организации и проведения практики для получения первичных профессиональных навыков часть 1 колледже электроники и бизнеса ОГУ для студентов 2 курса 4 семестра по специальности 220105 “Программное обеспечение вычислительной техники и автоматизированных систем” очной формы обучения.

Методические указания составлены с учетом Государственного Образовательного стандарта среднего профессионального образования по направлению подготовки дипломированных специалистов – утвержденного 08.02.2002 Министерством Образования Российской Федерации.

ББК 22.18я73

Содержание

Общие положения

Практика является одной из основных форм учебного процесса, направленная на формирование и воспитание высококвалифицированного специалиста.

Определяющим учебно-методическим документом практики, регламентирующим учебную деятельность студента и деятельность преподавателя на практиках, являются методические указания по производственной практики.

Организация практики осуществляется кафедрой - вычислительной техники и математики. Общая координация работ по организации производственной практики осуществляется учебным отделом колледжа.

Непосредственное участие в подготовке и проведении практики принимает декан факультета через своего ответственного, назначаемого из числа преподавателей кафедры ВТ и М.

Руководители практики от кафедры:

- обеспечивают проведение всех организационных мероприятий перед началом практики (инструктаж о порядке прохождения практики, по технике безопасности и т.д.)
- обеспечивают высокое качество прохождения практики студентами и строгое соответствие ее учебным планам и программам.
- рассматривают отчеты студентов по практике, дают отзывы по их работе и представляют заведующему кафедрой письменный отчет о проведении практики вместе с замечаниями и предложениями по совершенствованию практической подготовки студентов;
- обучает студентов-практикантов безопасным методам работ;

1 Обязанности студента

Студент при прохождении практики обязан:

- полностью выполнять задания, предусмотренные программой практики,
- подчиняться действующим в колледже правилам внутреннего трудового распорядка,
- изучить и строго соблюдать правила охраны труда, техники безопасности и производственной санитарии,
- вести дневник, в который записывать необходимые сведения, содержание лекций и бесед, делать эскиз, зарисовки и т. д.;
- представить руководителю практики письменный отчет о выполнении всех заданий и сдать зачет по практике.

2 Теоретические занятия и производственные экскурсии

В дни теоретических занятий студенты слушают лекции и беседы, участвуют в экскурсиях, получают консультации, собирают материал по программе практики, выполняют индивидуальные задания всех разделов практики, работают с технической документацией

3 Отчет о практике

По окончании практики студент составляет письменный отчет и сдает его руководителю практики от кафедры.

Содержание отчета определяется видом проходимой студентом практики, в данном случае он должен включать итоги и выполнения индивидуального задания и специальных вопросов. Перечень этих вопросов указывается и пунктах «Отчетность по практике» для каждого вида практики. Кроме того, в отчете должен быть краткий анализ работы производственного участка, на котором студент провел основное время практики, выводы и предложения по совершенствованию этой работы. В отчете должна быть использована необходимая производственная документация. Общий объем отчета 20—25 страниц. Для оформления отчета студенту выделяется в конце практики 2—3 дня.

По окончании практики студент сдает зачет (защищает отчет) с дифференцированной оценкой .

При оценке итогов работы студента на практике принимается во внимание характеристика, данная ему руководителем практики от предприятия.

4 Цель и задачи практики, ее место в учебном процессе.

Целью практики является приобретение студентами опыта алгоритмизации, конструирования и практической реализации программ на ЭВМ с использованием современных алгоритмических языков программирования.

В процессе 1 этапа практики для получения первичных профессиональных навыков студенты должны разработать обобщенную схему алгоритма по словесному описанию задачи, провести детализацию отдельных блоков обобщенной схемы, выделить необходимые процедуры и функции, определить наборы логически связанных между собой данных (потоки данных), ввести различные дополнительные средства для обеспечения наглядности и повышения уровня сервиса проектируемой программы, разработать и отладить программу, реализующую спроектированный алгоритм, выполнить на ЭВМ сконструированную программу и оформить программные документы в соответствии с требования ЕСПД.

Для успешного выполнения практики получения первичных профессиональных навыков студент должен обладать подробными сведениями о работе в операционной системе, уметь пользоваться библиотеками Турбо Паскаля, владеть знаниями особенностей алгоритмического языка .

Практика для получения первичных профессиональных навыков базируется на курсах "Информатика", "Основы Алгоритмизации и программирование", "Операционные системы и среды" и проводится на 2-ом курсе в 4 семестре - по завершению указанных предметов.

5 Содержание практики для получения первичных профессиональных навыков.

1 часть практики - 3 недели (январь- февраль месяц). В ходе практики студент должен изучить материал по предмету «Основы алгоритмизации и программирования» по темам: модули CRT, GRAPH и выполнить индивидуальные задания по данным темам а именно:

1) разработать программу по теме: «массивы» с использованием процедур и функций текстового режима (стандартный модуль CRT,)

2) Изучить процедуры и функции модуля GRAPH

3) разработать эскиз графического объекта и программу к этому эскизу используя все возможности модуля GRAPH

4) разработать техническую документацию на программу

5) оформить документацию в текстовой редакторе Microsoft Word .

Отчет по практике оформляется в виде пояснительной записки.

Завершается практика защитой отчета.

6 Семинарские занятия по практике

1) Установочные занятия. Цель и ход практики. Выдача заданий.

2) Разработка алгоритмов программ по теме: "Стандартные модули CRT, GRAPH и их процедуры и функции".

3) Проектирование программ и написание спецификаций.

4) Документация программ.

7 Организационно-методические данные практики

Организационно методические мероприятия практики указаны в таблице 1 они предусматривают как лекционные так и лабораторно –практические занятия, общая трудоемкость практики 60 часов.

Таблица 1- Организационно методические мероприятия

Вид работы	Трудоемкость в часах
1.Аудиторная работа в том числе: а) лекции б) лабораторно-практичекие работы в) форма итогового контроля г) общая трудоемкость дисциплины	 12 48 зачет 60

7.1 Этапы выполнения практики для получения первичных профессиональных навыков:

В таблице 2 календарно-тематический план который содержит этапа практики наменование разделов и тем лекционных и лабораторно- практических занятий.

Таблица 2- Календарно-тематический план

№ Занятий	Наименование разделов и тем	Количество часов	
1	2	3	
1	**Задачи практики**:		
1.1	Стандартизация документации- отчет по практике		
1.2	Техника безопасности и охрана труда		
1.3	Выдача задания на практику.	4	1 неделя
2	**Модуль CRT**. Установка цветом символа (изображе-		
	ния).Процедуры действующие в окнах (Window)	4/8	
3	Составление алгоритмов и программ с использованием	4/12	
	массивов. Выбор в массиве по критериям	4/16	
4	Отладка программы По индивидуальному заданию	2	
	Контрольный просмотр	2/20	
5	Оформление программы по индивидуальному задания		2 неделя
	с использованием процедур модуля CRT	4/24	
6	Оформление выполненных домашних работ	4/28	
7	***Контрольный просмотр***	4/32	
8	Модуль GRAPH. Стандартные процедуры и функции	4/36	
	модуля.		
9	Работа с текстом в модуле GRAPH. Переключение	4/40	
	между текстовым и графическим режимом	4/44	
10	Разработка эскиза и алгоритма индивидуального	4/48	3 неделя
	задания.		
11	Разработка программы.	4/52	
12	Отладка и редактирование программы индивидуаль-	2/54	
	ного задания		
	Контрольный просмотр	2/56	
13	Составление программы с использованием процедуры		
	OutText, OutTextXY		
14	***Подготовка к защите выполненных работ***	4/60	
	Итого:	60ч.	

8 Требования к построению отчета по практики

8.1 Структурные элементы отчета по практики

Отчет по практики для получения первичных профессиональных навыков должна содержать:

- титульный лист;
- задание;
- содержание;
- введение,
- основную часть(пояснительную записку);
- список использованных источников;
- заключение;
- приложения.

8.2 Титульный лист

Титульный лист является первым листом отчета. Переносы слов в надписях титульного листа не допускаются. На титульном листе указывают классификационный код. Титульный лист оформляют в соответствии с приложением А.

8.3 Задание

Задание на практики для получения первичных профессиональных навыков могут быть индивидуальными или комплексными, но примерно одинаковы по объему и степени сложности поставленных перед студентами задач. Тематика заданий практики рассматривается и принимается кафедрой, а утверждается заместителем. директора по учебной работе. Бланк задания оформляют в соответствии с приложением Б.

8.4 Содержание

Содержание включает введение, порядковые номера и наименование разделов и подразделов с указанием номеров листов, список использованных источников и приложение.

Слово «Содержание», записывают посередине страницы с прописной буквы. Пример оформления содержания приведён в приложении Г.

8.5. Введение. Заключение

Введение - вступительный раздел основного текста отчета по практики. Цель введения - познакомить с существом вопроса. В этом разделе указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие. Введение не включается в общую нумерацию разделов и

размещают на отдельном листе. Введение - не обязательный структурный элемент. В заключении содержатся выводы и рекомендации относительно возможностей практического применения материалов проекта.

Слова «Введение» и «Заключение» записывают посередине страницы с прописной буквы.

8.6 Список использованных источников

Список должен содержать сведения об источниках, использованных при составлении курсового проекта.

Наименование структурного элемента «Список использованных источников» записывают посередине страницы с прописной буквы.

Ссылки на источники следует указывать порядковым номером по списку источников, выделенным двумя косыми чертами.

Использованные источники нумеруют арабскими цифрами и располагают в порядке появления ссылок на источники в тексте.

8.7 Приложения

Материал, дополняющий текст отчета по практике, допускается помещать в приложениях. Приложениями могут быть, например, таблицы большого формата, графический материал, структурные схемы и т.д.Приложения оформляют как продолжение данной работы на последующих его листах.

Каждое приложение следует начинать с нового листа с указанием наверху посередине страницы слова «Приложение» и его обозначение.

Приложения обозначают заглавными буквами русского алфавита, начиная с А, за исключением Е, 3, Й, О, Ч, Ь, Ы, Ъ.

В тексте отчета по практике должны быть ссылки на все приложения.

9 Методические указания

Система программирования Turbo Pascal называется интегрированной средой программирования, так как она объединяет в себе возможности ранее разрозненных средств, используемых при разработке программ: редактора текста, компилятора, компоновщика, отладчика, и при этом обеспечивает программисту великолепные сервисные возможности.

Интегрированная среда программирования Turbo Pascal имеет следующие возможности:

1) множество накладывающихся окон
2) поддержка мыши, меню, диалоговых окон
3) много файловый редактор, который может редактировать файлы до 1Мб
4) расширенные возможности отладки
5) полное восстановление и сохранение среды разработки
6) множество стандартных модулей.

Управление цветом символов (изображения) в текстовом режиме. Возможности по управлению цветом выводимого символа зависит от режима работы дисплея. В графическом режиме цвет символа (зависит) совпадает с цветом переднего плана; цвет фона, окружающего символом, совпадает с цветом фона

Color [передний план] [[фон][окаймление]]

где-

Передний план – числовые выражения которого задает цвет символа - совокупность точек, формирующих само изображение.

Например, если на экране изображена буква А, то передний план – это точки, составляющие букву А, а фон – точки, окружающие букву А.

Как правило, изображение формируется положением точек переднего плана на фон.

Фон(цвет) – числовое выражение в пределах от 0 до 7, в которой окрашиваются точки экрана, если номер цвета равен 0, он не фиксирован и может быть любым из цветов, поддерживаемых дисплеем.

Окаймление – числовое выражение (только для графики)- область расположенная за пределами области экрана, в которой формируется изображение. Как правило оно окрашено в тот же цвет, что и фон, и поэтому неотличимо от фона. Эта область не участвует в формировании изображении; она нужна для повышение качества изображения за счет развертки за пределами рабочей области экрана.

Для цветного дисплея номера цветов от 0 до 7 составляют основной набор цветов, а номера от 8 до 15 светлые оттенки этого набора в таблице 3 представлены номера цветов в текстовом режиме. Базовые цвета: синий, зеленый, красный.

Таблица 3- Номера цветов в текстовом режиме.

Номер цвета	Цвет	Номер цвета	Цвет
0	Черный	8	Серый
1	Синий	9	Светло-синий
2	Зеленый	10	Светло-зеленый
3	Голубой (сине-зеленый)	11	Светло-голубой
4	Красный	12	Розовый (светло-красный)
5	Пурпурный (красно-синий)	13	Светло-пурпурный
6	Коричневый (красно-зеленый)	14	Желтый (светло-коричневый)
7	Белый (сине-зелено-красный)	15	Ярко-белый

9.1 Модуль CRT. Текстовый режим использования экрана.

Стандартный модуль CRT является важнейшей частью интегрированной среды Turbo Pascal. Он устанавливает режим работы адаптера дисплея, организует прямой вывод в буфер экрана, регулирует яркость свечения символов, выполняет многие другие жизненно важные функции, необходимые для

нормальной работы компьютера. Курсор в этом режиме можно переводить в любую позицию (знакоместо) экрана.

Можно выделять цветные прямоугольники (поля, окна)

Активным всегда является одно окно:

- но содержание других окон может сохраняться на экране;
- созданное прежде окно может вновь стать активным.

По умолчанию окном является весь экран. Наглядность может дополняться “музыкальным сопровождением.”

Процедуры входящие в эту группу, позволяют программисту управлять состоянием строк на экране и выполнять другие действия, относящиеся к работе с экраном:

- Clrscr - полностью очищает экран и помещает курсор в левый верхний угол экрана (координаты 1;1) выполняется только в текстовом режиме работы экрана
- Deline - полностью стирает содержимое строки, в которой расположен курсор
- Delay - выполняет задержку на заданное число миллисекунд
- GotoXY(x,y) - позиционирует курсор в точку с заданными координатами X,Y
- Keypressed - возвращает значение True, если на клавиатуре нажата клавиша и False в противном случае
- Sound(x) - включает внутренний динамик, (X) указывает частоту звучания в герцах
- Nosound - выключает внутренний динамик
- ReadKey - считывает символ с клавиатуры
- TextBackGround - устанавливает фоновый цвет
- TextColor - выбирает цвет символа
- Window(x1,y1,x2,y2)определяет на экране текстовое окно,.

9.1.1 Управление звуком

Для создания звуковых эффектов в Турбо Паскале используются стандартные процедуры *Sound, NoSound* и *Delay* модуля CRT.

Sound(I: word) — активизирует звуковые средства персонального компьютера. Целочисленное значение I указывает частоту звучания звука в герцах. Звук указанной частоты будет генерироваться до тех пор, пока не будет отменен процедурой *NoSound.*

NoSound -отмена звука. Отменяет звуковой режим, заданный процедурой Sound.

На некоторых типах персональных компьютеров процедура *Sound* аппаратно не поддерживается и реализация звуковых эффектов невозможна.

Для указания времени, в течение которого будет продолжаться звучание, используется процедура *Delay*.

Пример 1. Фрагмент к программе

..... begin Sound(500); Delay(2000); NoSound end;……

В данном примере звуковой сигнал частотой 500 Гц будет звучать в течение 2 с (2000 мс).

С помощью процедур *Sound, NoSound, Delay* и операторов цикла можно создать самые разнообразные звуковые эффекты: звучание сирены, метро-нома, будильника, пение птиц, фрагменты музыкальных произведений и т.д. Для этого используется набор частот или элементы массива, соответствующие нотам различных октав В таблица 4 представлены набор частот соответствующие нотам различных октав.

Таблица 4- Набор частот соответствующие нотам различных октав

Нота	Большая октава	Малая октава	Первая октава	Вторая октава
До	130.81	261.63	523.25.	1046.50
Ре	146.83	293.66	587.33	1174.07
Ми	164.81	329.63	659.26	1318.05
Фа	174.61	349.23	698.46	1396.09
Соль	196.00	392.00	784.99	1568.00
Ля	220.00	440.00	880.00	1760.00
Си	246.94	493.88	987.77	1975.00

Для использования в процедуре *Sound* все указанные в таблице значения частот округляются.

Пример 2.Генерация мелодий

Способ построения мелодичных звуковых рядов с использование частот, соответствующих нотам. Частоты загружаются в один массив, продолжительность звучания каждой частоты - в другой. В примере представлена процедура, генерирующая гамму с нарастающей продолжительностью звучания каждой ноты.

```
procedure  Gamma;                    {Ноты}
  const M:array[1..7]  of  integer  (262,294,330,349,392,440,494);
                                     {Продолжительность  звучания}
   T:  array[1..7]  of  integer  =(10,11,12,13,14,15,16);
    Var I:  byte;
    begin
           while not KeyPressed do
     begin
      for I:= 1 to 7 do begin Sound(M[I]); Delay(T[I]); NoSound  end ;
     end;
   end ;
```

Меняя значения элементов массивов *M* и *T*, можно добиться довольно хорошей имитации музыкальных произведений.

Пример 3 Программа, позволяющая воспроизвести простую музыкальную гамму (частоты всех полутонов первой октавы записаны в массив *F*).

```
program gamma;
 uses Crt;
 const F : array [1..12] of real =(begin
   for K:=0  to  3  do
   for N:=l   to   12  do
    begin
        Sound(Round(F[N]*(1   shl   k   ))); Delay(Temp); NoSound;
         end ;
  for  K:=3  downto  0  do
   for N:=12  downto  1  do
   begin
        Sound(Round(F[N]*(1   shl   к   ))); Delay(Temp); NoSound;
        end;
     end.
```

На основе этого примера вы можете создавать фрагменты программ, обеспечивающие звуковое сопровождение ваших программ. Обратите внимание на то, что при переходе от одной октавы к соседней частоты изменяются в два раза.

Пример 4. Звуковое сопровождение процесса вывода

Весьма впечатляет вывод информации, сопровождаемый звуковыми эффектами. Он применяется при формировании заставок, подсказок, сообщений об ошибках и т. д. Например, пусть требуется вывести в центр экрана элемент заставки

“ЭКСПЕРТНАЯ СИСТЕМА”

в сопровождении звукового сигнала. Генерацию сигнала оформить в процедуре *Zvuk*:

```
procedure  ZvukOut;
 var St: String[40]; I: byte;
   procedure  Zvuk;
   begin
           Sound(5000);
             Delay(70);
               NoSound
   end; {Zvuk}
begin
  ClrScr;
            St:=   'ЭКСПЕРТНАЯ       СИСТЕМА';
GoToXY(18,12);
 for  I:=  1   to Length(St)   do
```

```
    begin
          Write(St[I]);
                Zvuk
      end
end;
```

Кроме сопровождения вывода строковых и любых других данных в ряде типов программ (например, игровых) требуется формирование более сложных звуков: криков животных, шума автомобильных или авиационных двигателей и т.д. Для организации подобных имитаций требуется определенный навык и опыт работы.

Звуковое сопровождение этапов выполнения программы применяется как дополнительное средство для контроля решения задачи на персональном компьютере. Во время длительного счета на одной машине пользователь может работать на другом компьютере и следить за работой первой по звуковым сигналам. Звуковое сопровождение может носить самый разнообразный характер: от определенного количества элементарных сигналов типа ^G до имитации человеческого голоса.

Пример 5: Программа, в которой после выполнения каждого из трех блоков на экран выводится сообщение и выдается один или несколько звуковых сигналов. Количество сигналов равно номеру отработанного блока, program DemoSignal;

```
   ……..
procedure   Signal(N:byte);
 var I: byte;
 begin
          for I:= 1 to N do
begin
       Write(G);  Delay(500);
End;
 end;
  begin            {Основная программа}
      Write('Отработал блок 1');
       Signal(1);
      Write('Отработал блок 2’);
       Signal(2);
      Write('Отработал блок 3');
        Signal(3);
  end.
```

Заменив в процедуре *Signal* оператор *Write(^G)* на имя процедуры, генерирующий звуковой ряд, можно добиться повторения его звучания нужное количество раз.

Пример 6 . Изучите программу *Dem_Music*, которая имитирует разные звуки.

```
program Dem_Music;
 uses  Crt;
```

{Процедура генерации звуков *Bird* получает из основной программы три параметра-значения. В зависимости от селектора *KindOfBird*, которому при вызове процедуры *Bird* присваивается целочисленное значение от 1 до 4, а также параметров-значений *Duration* и *Pitch*, генерируются различные звуки}

```
procedure Bird(KindOfBird:Byte;Duration:Word; Pitch: integer);
  var I, J, K : Integer;
begin I:=1;
    case KindOfBird of
   1 : begin
       while I < Duration do
                begin Sound(Pitch + I) ; Inc (I);NoSound; end;
        begin
                    J := I + (I div 10);
       while I < Duration do
              begin
                   Sound(Pitch + i); Inc(I);
              end ;
                     NoSound;Delay(100);
       while I > J do
               begin
                 Sound(Pitch + I); Dec (I);
               end ;
                        NoSound;
end ;
        3 : begin
               K := 1;
                while K < Random(20) + 10 do
                 begin
                      Sound(Pitch);
                       Delay(30);
                    Sound(Pitch + (Pitch div 10)) Delay(30); Inc(K);
                end ;
                       NoSound;
                end;
        4 : begin
               I := Random(5) + 10; K := Pitch + (Pitch div 10);
                while I > 1 do
                begin
                     J := Pitch; while J < K do
                      begin
                        Sound(J);
                         Delay(6); Inc(J, 10) ;
                         end ;
                    NoSound;  Delay(40); Dec(I);
                       end ;
```

```
            end;
       end;          {Case}
    end;
```

{Основная программа. Пока не нажата любая клавиша, выполняется вызов процедуры *Bird* с параметрами-значениями, величина которых задается с использованием функции *Random*. После нажатия на любую клавишу цикл завершается, звуковой режим отменяется, и программа завершает работу.}

```
  ……..
  begin
    Randomize;
    repeat      {Повторять, пока не нажата любая клавиша}
   Bird(Succ(Random(4)),Random(2000)+300,Random(4000)+ 200);
    Delay(Random(400) + 20);
     until KeyPressed;
    NoSound;
    end.
```

Обратите внимание на применение стандартных процедур *Sound*, *NoSound*, *Delay*, *Dec*, *Inc*, функций *Random*, *Succ*. Запустите интегрированную среду программирования, введите текст программы Dem_Music и запишите файл на диск под соответствующим именем, а затем откомпилируйте его и проверьте действие программы.

9.1.2 Процедура WINDOW. Процедуры, действующие в окнах.

1.Процедура Window активирует окно, определяя его границы.
Формат записи процедуры - *Window(x1,y1,x2,y2)*
Параметры процедуры x2>x1, y2>=y1

2.Цвет фона задает процедура *TextBackGround(<Цвет>)*

Чтобы получить на экране цветное поле, нужно обратиться к двум процедурам и вслед выполнить очистку окна.

Пример 7: построения окна синего цвета расположенного в левом верхнем углу экрана.

```
……
TextBackGround(1); {Синий цвет окна}
Window(1,1,15,10); {Окно в левом верхнем углу экрана}
ClrScr;              {Очищаем окно – курсор в левом верхнем углу экрана}
……
```

3.Цвет символа задает процедура *TextColor(<Цвет символа>);*

Пример 8: Фрагмент меню- названия выводятся на синем фоне светло-серым цветом, а номера пунктов ярко-белым.

```
……
TextBackGround(Blue);
   TextColor(15);   Write('4');
  TextColor(7);    Writeln('Вывод на экран');
   TextColor(15);   Write('2');
 TextColor(7);    Writeln('Вывод на принтер');
……..
```

Функция

-*Where X* – возвращает относительную горизонтальную координату текущей позиции курсора.

-*Where Y* – возвращает относительную вертикальную координату текущей позиции курсора.

9.2 Модуль GRAPH.

Модуль *Graph* реализует библиотеку более чем 50 графических программ. Поддерживает несколько типов и стилей заполнения. Существует несколько типов шрифтов, которые можно увеличивать, выравнивать и ориентировать горизонтально и вертикально. Экран представляет собой прямоугольное поле из точек. Цвет каждой точки можно изменять. Экран может находиться либо в текстовом, либо в графическом режиме. Видеопамять, контроллер лучевой трубки, порты ввода/вывода и т.д., располагаются на печатной плате – адаптере.
Существует несколько типов адаптеров:

1.С разрешающей способностью экрана, которую они в состоянии поддерживать.

2.С максимальным количеством цветов одновременно на экране.

Графическую настройку выполняют графические драйверы, которые располагаются в файлах с расширением .BGI (Borland Graphics Interrface).

Выпускаемые в настоящее время ПК оснащаются адаптерами , разработанными фирмой IBM, или совместимыми с ними. Они имеют возможность работы в графическом режиме. В этом режиме экран дисплея рассматривается как совокупность точек – пикселей, светимость которых можно управлять с помощью программы. Графические возможности конкретного адаптера определяются разрешение экрана, т.е.общим количеством пикселей, а также количеством цветов(оттенков), которыми может светиться любой из них.

9.2.1 Как перевести экран в графический режим?

Для этого используется процедура модуля *Graph InitGraph*; Она определяет графическое устройство, загружает и инициализирует соответствующий графический драйвер, переводит систему в графический режим и возвращает

управление вызывающей программе.

Симметричной процедурой к *InitGraph* является процедура *Closegraph* – выгружающая драйвер из памяти и восстанавливает исходный видеорежим.

InitGraph(GD,GM,Path); *{Перевести экран в графический режим}*
GD – номер драйвера
GM – номер режима
Path – путь доступа к файлу, содержащему нужный драйвер.

Если *Path* содержит просто пустую строку (*Path:=''*), то драйвер ищется в текущем каталоге.

GD и *GM* – являются переменными параметрами.

Если при запуске *InitGraph* переменная *GD* равна 0, то драйвер и оптимальный графический режим для этого драйвера определяется автоматически.

Пример 9.

```
Program gr1;
    Uses Graph, Crt;
  Var Gd, Gm: Integer;
Begin
 Gd:= Detect;                  {автоматическое определение драйвера}
Initgraph (Gd,Gm,' Path' );  {инициализация графического режима}
……..
Readln;                       {ждать нажатия клавиши}
Closegraph;                   {процедура выгружает графический режим}
End.
```

MoveTo(x,y:Integer) – перемещает указатель в точку с координатами *x,y*
Moverel(dx,dy:Integer) – перемещает *CP* на *dx* точек по горизонтали и на *dy* точек по вертикали относительно последнего положения текущего указателя «+» - увеличивает; «-» уменьшает.

9.2.2 Обработка ошибок при инициализации графического режима

Графическая программа, как и любая другая, может содержать ошибки. Программист должен предусмотреть всё возможное для их своевременного обнаружения и нейтрализации. Для этого в модуле Graph имеются функция *GraphResult.*

Заголовок функции:

Function GraphResult: Integer

Эта функция, при обращении к ней, возвращает специальный код ошибки (целое число), в зависимости от того, как прошло выполнение любой графической

процедуры или функции. Если ошибка не обнаружена, то значением функции будет ноль, в противном случае - отрицательное число. Все возможные ошибки и их коды приведены в таблице 5.

Для того чтобы выводить сообщение об ошибке используется функция *GraphErrorMsg.*

Заголовок функции:

Function GraphErrorMsg(Code: Integer): String

Данная функция возвращает значение типа *String,* в котором по указанному коду дается соответствующее текстовое сообщение об ошибке (по-английски).

Здесь *Code* - код ошибки, возвращаемый функцией *GraphResult*

Таблица 5- Константы и коды ошибок графических операций

Имя константы	Код ошибки	Соответствующее сообщение об ошибке (перевод)
grOk	0	Нет ошибок
grNoInitGraph	-1	Не инициализирован графический режим
grNotDetected	-2	Не удалось определить тип адаптера
grFileNotFound	-3	Не обнаружен файл с драйвером
grInvalidDriver	-4	Неправильный тип драйвера
grNoLoadMem	-5	Нет памяти для размещения драйвера
grNoScanMem	-6	Нет памяти для просмотра областей
grNoFloodMem	-7	Нет памяти для закраски областей
grFontNotFound	-8	Не найден файл со шрифтом
grNoFontMem	-9	Нет памяти для размещения шрифта
grInvalidMode	-10	Неправильный графический режим
grError	-11	Общая ошибка
grIOError	-12	Ошибка ввода-вывода
grInvalidFont	-13	Неправильный формат шрифта
grInvalidFontNum	-14	Неправильный номер шрифта

Пример 10: программа которая производит инициализацию графического режима и проверку возможных ошибок:

```
uses graph;                          {Подключение модуля Graph}
   var gd,gm,error:integer;          {Раздел описания переменных}
begin                                {Начало программы}
   gd:=detect;                       {Автоматическое определение драйвера}
InitGraph (gd,gm,");                 {Инициируем графику}
   error:=GraphResult;               {Получаем результат}
if error<>grok then                  {Проверяем ошибку}
   WriteLn (GraphErrorMsg(Error))    {Выводим сообщение}
```

```
else                    {Нет ошибки}
begin                   {Начало основной части программы}
.....                   {Основная часть программы}
Readln;                 {Ожидание нажатия клавиши Enter}
CloseGraph;             {Закрытие графического режима}
end;                    {Конец основной части программы}
end.                    {Конец программы}
```

Такая проверка должна проводиться в каждой программе, в которой используется графический режим (именно для этого предназначена функция *GraphResult).*

9.2.3. Переключение между текстовым и графическим режимами

Иногда требуется периодически переходить из текстового режима в графический и наоборот. Для того, что бы из текстового режима перейти в графический режим работы нужно использовать процедуру *SetGraphMode.*

Заголовок процедуры:

Procedure SetGraphMode (Mode: Integer)

Эта процедура устанавливает новый графический режим работы адаптера. Здесь *Mode* - код устанавливаемого режима.

Для того, что бы перейти из графического режима работы в текстовый режим работы необходимо использовать процедуру *RestoreCRTMode.*

Заголовок процедуры:

Procedure RestoreCRTMode

Эта процедура восстанавливает видеорежим, который был до инициализации графического режима. Данная процедура не освобождает память и не сбрасывает параметры графического режима. Процедуру *RestoreCRTMode* можно использовать совместно с процедурой *SetGraphMode,* чтобы временно (и быстро) переходить в программе из графического режима в текстовый и обратно.

Пример 11: Программа иллюстрирует переход из графического режима в текстовый и обратно:

```
Uses graph;
        Var gd, gm, error: integer;
Begin
      gd:=detect;
       InitGraph (gd,gm,");
        Error:=GraphResult;
If error<>grok then WriteLn (GraphErrorMsg(Error))
        Else
```

```
  begin
        WriteLn ('Это графический режим работы');
        WriteLn ('Нажмите клавишу "Enter"...':20);
        ReadLn;
                                          {Переходим в текстовый режим}
        RestoreCRTMode;
             WriteLn (' А это текстовый режим работы');
            WriteLn ('Нажмите клавишу "Enter"...':20);
        ReadLn;
                                        {Возвращаемся в графический режим}
         SetGraphMode (GetGraphMode);
            WriteLn ('Опять графический режим...');
         ReadLn;
           CloseGraph;
    End;
End.
```

9.3. Описание графических процедур и функций

9.3.1. Система координат

Для того чтобы мы могли что-либо изобразить на экране, мы должны использовать систему координат (см. рис. 2.1). Отсчёт начинается от верхнего левого угла экрана, который имеет координаты (0,0). Привычная для нас система координат "перевёрнута". Каждая точка на экране на самом деле представляет собой очень маленький прямоугольник (и поскольку это не совсем точка, то иногда используют специальный термин - "пиксель"). Количество точек (пикселей), умещающихся на экране по вертикали и горизонтали, называют разрешающей способностью. Разрешающая способность экрана в режиме VGAhi - 640x480. Это означает, что по горизонтали на экране умещается 640 точек, а по вертикали - 480.

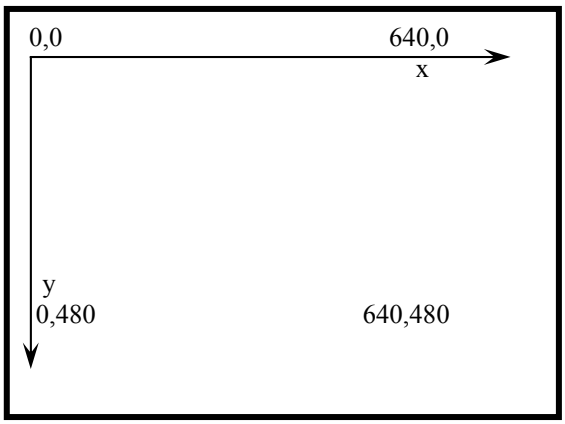


Рисунок 1- Координаты экрана в режиме VGAhi

9.3.2. Текущий указатель

Чтобы строить изображение, необходимо указывать точку начала вывода. В текстовом режиме эту точку указывает курсор, который виден на экране, если его конечно не убрать. В графическом режиме видимого курсора нет, но есть невидимый текущий указатель CP(Current Pointer). Фактически это тот же курсор только он невидим.

Рассмотрим процедуры и функции позволяющие перемещать указатель текущей позиции в любую точку экрана и определяющие где указатель в данный момент находиться.

Заголовок процедуры:

Procedure MoveTo(x, y: Integer)

Эта процедура перемещает указатель текущей позиции в точку с заданными координатами, где x,y - координаты точки на экране, в которую будет перемещен указатель. Точка на экране не высвечивается.

Отсчет координат ведётся относительно верхнего левого угла экрана или если есть окно, то относительно верхнего левого угла окна.

Заголовок процедуры:

Procedure MoveRel (dx, dy: Integer)

Данная процедура перемещает указатель на заданное расстояние относительно его текущего местоположения, где dx, dy - приращения текущих координат x и y.

Например, если первоначальное положение текущего указателя было в точке с координатами (0,0), то процедура *MoveTo*(100,300) переместит указатель в точку с координатами 100, 300, а процедура *MoveRel*(5,10) переместит указатель из точки с координатами 100, 300 в точку с координатами 105, 310.

В некоторых программах приходиться выполнять постоянный контроль местоположения текущего указателя. Для этого используются следующие функции.

Заголовок функции:

Function GetX:Integer

Эта функция возвращает координаты по горизонтали указателя текущей позиции.

Заголовок функции:

Function GetY:Integer

Эта функция возвращает координаты по вертикали указателя текущей позиции.

Координаты *GetX* и *GetY* отсчитываются относительно левого верхнего угла

активного окна, или, если окно не установлено, экрана.

Для того чтобы узнать максимально возможные для установленного режима значения координат X и Y используют функции.

Заголовки функций:

Function GetMaxX:Integer
Function GetMaxY:Integer

Данные функции возвращают текущие координаты разрешений (т.е. число пикселей) по горизонтали и вертикали соответственно. Также с помощью этих функций можно определить координаты центра экрана.

```
Xcenter:=GetMaxX div 2;
Ycenter:=GetMaxY div 2;
```

Такой способ определения центра экрана избавляет от настройки на конкретный тип монитора.

9.3.3. Окно в графическом режиме

Также как в текстовом режиме графический экран можно рассматривать как одно большое окно или несколько маленьких окон. Вывод можно осуществлять не только на весь экран, но и в пределах одного окна. Напомним что окно - это прямоугольная область экрана, выполняющая все функции полного экрана. Если окон несколько, то в каждый отдельный момент может быть активно только одно окно. Рассмотрим несколько процедур предназначенные для работы с окнами в графическом режиме.

Для того чтобы создать окно в графическом режиме следует воспользоваться процедурой.

Заголовок процедуры:

Procedure SetViewPort(x1, y1, x2, y2: Integer; Clip: Boolean)

Где (x1,y1) - координаты, левого верхнего угла, (x2,y2) - координаты правого нижнего угла окна. Параметр *Clip* определяет, должно ли отображаться на экране то, что не помещается в окне. Параметр *Clip* может принимать два значения - *True* и *False.* Эти значения можно задавать, используя две определенные в модуле константы:

```
Const
  ClipOn = True; {Включить отсечку}
  ClipOff = False; {Не включать отсечку}
```

Если параметр *Clip* имеет значение ClipOn, то часть изображения не помещающаяся в окне, обрезается по его краю. Если параметр *Clip* имеет значение ClipOff, то изображение будет представлено на экране полностью на рисунке 2.

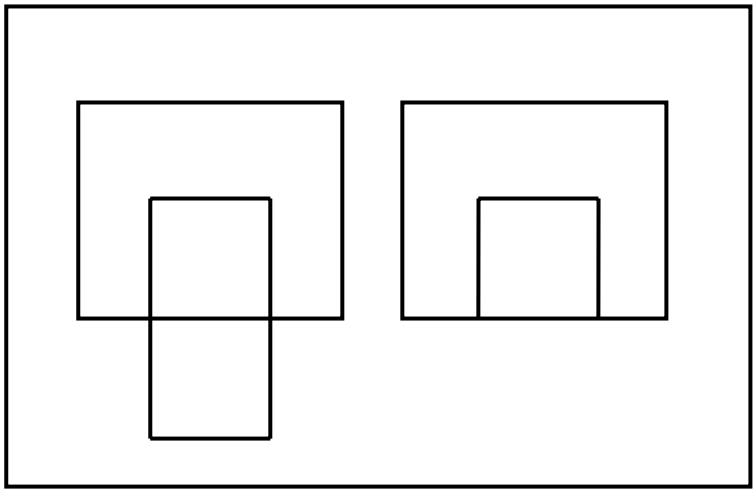


Рисунок 2 - Действие параметра *Clip*

На рисунке изображены два окна с разными значениями параметра *Clip.* Для левого окна *Clip* равен ClipOff, а для правого ClipOn. Изображение прямоугольника начинается в окне и заканчивается за его пределами. Для первого окна (ClipOff) прямоугольник отображается полностью, для второго(ClipOn) – обрезается по его краю.

После того как окно создано, координаты его верхнего левого угла становятся равны (0,0).

Для процедур и функций, что-либо отображающих, перемещающих текущий указатель или выясняющих местоположение указателя координаты отсчитываются относительно активного окна.

Чтобы стереть все изображения на экране, т.е. очистить его, используется процедура ClearDevice.

Заголовок процедуры:

Procedure ClearDevice

Данная процедура не имеет параметров. Она очищает экран и перемещает текущий указатель позиции в левый верхний угол, т.е. в точку с координатами (0,0). При очистке экран заполняется цветом фона, установленным процедурой SetBkColor(см ниже).

Для очистки окна используется процедура ClearViewPort.

Заголовок процедуры:

Procedure ClearViewPort

Данная процедура не имеет параметров. Она очищает активное окно. Если на экране окна не создавались, то очищается весь экран. При этом фон заполняется цветом из текущей палитры, код которого равен нулю, а текущий указатель

помещается в верхний левый угол.

9.3.4. Цвета, палитры, заполнения

Вывод изображений на экран в Turbo Pascal может осуществляться с использованием различных цветов. Набор цветов, доступных в том или ином режиме, называется палитрой. Кроме того, если речь идёт о заполнении экрана или какой-либо замкнутой фигуры, то это можно осуществить как сплошным фоном, так и некоторым узором. Рассмотри процедуры и функции, предназначенные для выше описанных видов действий.

Установить текущий цвет позволяет процедура SetColor.

Заголовок процедуры:

Procedure SetColor (Color: Word)

Данная процедура устанавливает текущий цвет - один из цветов используемой палитры. *Color* - новый текущий цвет. Значения, которые может принимать параметр *Color,* определяется набором определённых в модуле Graph констант. Перечень этих констант, а также соответствующих кодов можно посмотреть в таблице 6.

Таблица 6 - Константы и коды цветов

Имя константы	код	цвет
Black	0	Чёрный
Blue	1	Синий
Green	2	Зелёный
Cyan	3	Голубой
Red	4	Красный
Magenta	5	Фиолетовый
Brown	6	Коричневый
LightGray	7	Светло-серый
DarkGray	8	Тёмно-серый
LightBlue	9	Светло-синий
LightGreen	10	Светло-зелёный
LightCyan	11	Светло-голубой
LightRed	12	Розовый
LightMagenta	13	Малиновый
Yellow	14	Жёлтый
White	15	Белый
Blink	color + 128	Мерцание символа

Например, обращение SetColor(Yellow) сделает текущим цвет жёлтый, или

SetColor(2) сделает текущим цвет зелёный. Обращение можно осуществлять как через имя константы, так и через код.

Для того чтобы установить цвет фона для своего изображения необходимо применить процедуру SetBkColor.

Заголовок процедуры:

Procedure SetBkColor (Color: Word)

Данная процедура устанавливает цвет фона изображения - один из цветов используемой палитры. Параметр Color - определяет новый цвет фона.

Цвет фона определяется не для окна, а для всего экрана. Коды цветов фона могут находиться в диапазоне от 0 до 15, в зависимости от используемых драйвера и графического режима. Напомним, что в текстовом режиме для фона можно использовать только первую половину набора доступных цветов. Например, обращение SetBkColor(red) окрасит фон в красный цвет, или SetBkColor(5) - окрасит фон в фиолетовый цвет. Допускается использование в качестве параметра Color как имя константы, так и код.

В ряде случаев порядок следования цветов в стандартных палитрах может не удовлетворять программиста. Если требуется изменить расположение одного или нескольких цветов, можно воспользоваться процедурой SetPalette.

Заголовок процедуры:

Procedure SetPalette (Cut: Word; Paste: Shortint)

Данная процедура заменяет один из цветов в палитре новым цветом. Параметр Cut - код цвета в палитре (т.е. тот цвет, который будем менять), Paste - код нового цвета (т.е. тот цвет, на который меняют старый).

Процедура заменяет цвет, ассоциированный с кодом Cut, на цвет, код которого Paste. Например, обращение SetPalette(9,11) (или SetPalette(9,LightCyan), что то же самое) меняет цвет в палитре с кодом 9 (синий) на цвет с кодом 11 (светло - бирюзовый). После этого все, что было изображено на экране синим цветом, изменит свой цвет на светло-бирюзовый.

При построении изображения различные замкнутые области можно закрасить либо сплошным цветом, либо некоторым узором. Для этого используют процедуру SetFillStyle.

Заголовок процедуры:

Procedure SetFillStyle (Pattern: Word; Color: Word)

Данная процедура определяет шаблон и цвет заполнения. Параметр Pattern - определяет шаблон заполнения (см. таблицу), Color - цвет заполнения.

Заполнение - операция, позволяющая некоторый замкнутый контур закрасить сплошным цветом (например, цветом фона или текущим цветом) либо заполнить его некоторым узором. В модуле Graph имеется несколько готовых (стандартных) шаблонов заполнения, с каждым их которых ассоциирована мнемоническая константа. Перечень этих шаблонов представлен в таблице 7.

Таблица 7- Константы, коды и образцы заполнений

Имя константы	код	Заполнение
EmptyFill	0	Сплошное заполнение цветом фона (без узора)
SolidFill	1	Сплошное заполнение текущим цветом
LineFill	2	Заполнение символами --, цвет - color
LtSlashFill	3	Заполнение символами // нормальной толщины, цвет - color
SlashFill	4	Заполнение символами // удвоенной толщины, цвет - color
BkSlashFill	5	Заполнение символами \\ удвоенной толщины, цвет - color
LtBkSlashFill	6	Заполнение символами \\ нормальной толщины, цвет - color
HatchFill	7	Заполнение вертикально-горизонтальной штриховкой тонкими линиями, цвет - color
XHatchFill	8	Заполнение штриховкой крест-накрест по диагонали "редкими" тонкими линиями, цвет -color
InterLeaveFill	9	Заполнение штриховкой крест-накрест по диагонали "частыми" тонкими линиями, цвет -color
WideDotFill	10	Заполнение "редкими" точками
CloseDotFill	11	Заполнение "частыми" точками
UserFill	12	Шаблон заполнения, определённый пользователем

По умолчанию используются константа SolidFill(код 1) и цвет с максимальным номером из текущей палитры. Шаблон и цвет, определённые процедурой SetFillStyle, применяется всеми процедурами и функциями, в которых используется операция заполнения.

9.3.5. Отображение точки на экран

До сих пор речь шла о процедурах и функциях модуля Graph, настраивающих экран нужным образом перед выводом графических изображений. Теперь познакомимся со средствами, позволяющими что-то отображать на экране. В этом разделе мы рассмотрим процедуры и функции, выводящие на экран самые простые изображения - точки и линии.

Какие бы изображения не выводились на экран, все они построены из точек. Имея средство построения точки определённого цвета в нужном месте экрана, теоретически можно создать любое изображение. В библиотеке Graph вывод точки осуществляется процедурой PutPixel.

Заголовок процедуры:

Procedure PutPixel (x, y: integer; Color: Word)

Где *x,y*-экранные координаты точки

Color-ее цвет берется из установленной палитры.

Пользуясь оператором можно получить точечное изображение фигур.

```
…
PutPixel(159,99,2);
…
```

Пример 12: Построение окружности радиусом 50 с центром в точках 120, 90. точками.

```
 Program  krug;
            Uses crt;
              Var t: real;Gd,gm: integer;
 Begin
       Gd:=  detect;
         Initgraph(gd,gm,'    ');
    T:=0;
While t<6.28 do
 Begin Putpixel(round(120+50* cos(t)), round(90+50* sin(t)),2);T:=t+0.5;End;
Readln;
 Closegraph;
End.
```

Пример 13: В приведённом примере на экране изображаются 5000 точек. Координаты рисования точки и цвет точки выбирается случайным образом. Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

```
program tochki;
       uses crt,graph;
       var i,x,y,c:integer;
       gd,gm:integer;
begin
       clrscr;
         gd:=detect;
          initgraph(gd,gm, '   ');
   randomize;
for i:=1 to 5000 do
   begin
      x:=random(640);
      y:=random(480);
      c:=random(15);
               Putpixel(x,y,c);
      delay(50);
   end;
      readln;
   Closegraph;
      end.
```

9.3.6 Отображение отрезков прямых линий, построение прямоугольников.

Для отображение отрезков прямых линий используется следующие процедуры:

Line(x1,y1,x2,y2:Integer)

Где *x1,y1* – координаты начальной точки линии

x2,y2 – координаты конечной точки линии.

LineTo – строит отрезок из точки текущего положения указателя до точки с координатами *(x,y)*

LineRel(dx,dy) - проводит отрезок от точки текущего положения указателя до точки *(CPx+dx,CPy+dy)*,

где *Cpx,Cpy* – текущее положение *CP*.

Пример 14: построение ломаной линии

```
Program line;
  Uses graph, crt;
    Var t: real;Gd,gm:integer;
Begin
   Gd:= detect;
    Initgraph(gd,gm,' ');
  Line(20,80,80,80);Line(80,80,140,140); Line(100, 140,180,140);
  Readln;
    Closegraph;
End.
```

Пример15: В приведённом примере на экране изображается последовательность вертикальных линий. Чтобы получилась именно последовательность линий, а не сплошной прямоугольник, местоположение очередной линии относительно предыдущей сдвигается вправо на десять пикселей. В конце программы осуществляется задержка (чтобы можно было увидеть результат - ReadLn) и выход из графического режима(closegraph). Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

```
program linii;
uses crt,graph;
    var i: integer;
        gd,gm:integer;
begin
      clrscr;   gd:=detect;
     Initgraph (gd, gm,");
    For i:=1 to 50 do
        Line (50+i*10, 50, 50+i*10,400);
  readln;
       closegraph;
 end.
```

Пример 16: Цветные линии из центра экрана.

```
Program lin3;
    Uses graph, crt;
      Var x,y,c,gd,gm:integer;
Begin
       gd:= detect;
 Initgraph(gd,gm,'    ');
   SetBkcolor(15)
    Repeat
      x:=round(random(640));
       y:= round(random(470));
       c:= round(random(16));
   setcolor(c);
   Line(320,235,x,y);
 Until keypressed;
      Readln;
   Closegraph;
 End.
```

Для черчения линий применяются ещё две процедуры LineTo и LineRel.
Заголовок процедуры:

Procedure LineTo (x, y: integer)

Эта процедура отображает прямую линию от текущей позиции (точки, в которой находится указатель) до точки с заданными координатами. Параметры x и y - координаты точки. Например, если текущий указатель находиться в точке с координатами (100,100), то процедура LineTo(250,150) начертит линию, у которой начало будет в точке с координатами (100,100) а конец в точке с координатами (250,150).

С помощью процедуры LineTo линия отображается с использованием стиля и толщины, определённых процедурой SetLineStyle(см. далее); при этом используется цвет, заданный с помощью процедуры SetColor. Процедура LineTo очень удобна для вычерчивания различных многоугольников или ломаных линий.

Пример 17: программы использования процедуры LineTo:

```
program primer;
       uses crt,graph;
         var gd,gm:integer;
begin
      clrscr;
       gd:=detect;
       initgraph(gd,gm,");
          MoveTo(50,50);
   LineTo(200,100);
```

```
  LineTo(250,150);
       LineTo(150,250);
            LineTo(50,150);
               LineTo(50,50);
   readln;
           closegraph;
end.
```

В приведённом примере на экране изображается многоугольник. Первой в данной программе вызывается процедура MoveTo, которая перемещает указатель текущей позиции в заданную точку экрана. Затем последовательно пять раз вызывается процедура LineTo, которая каждый раз отображает очередную сторону нашего многоугольника. В конце программы осуществляется задержка (чтобы можно было увидеть результат - ReadLn) и выход из графического режима(CloseGraph). Чтобы многоугольник получился замкнутым, при вычерчивании последней стороны многоугольника процедуре LineTo должны быть переданы те же координаты, которые вначале были сообщены процедуре MoveTo. Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

Другая процедура LineRel.

Заголовок процедуры:

Procedure LineRel (Dx, Dy: integer)

Данная процедура отображает прямую линию от текущей позиции и до точки, координаты которой определены относительно положения указателя (т.е. текущей позиции). Параметры Dx и Dy - приращения координат текущей позиции x и y. Например, если указатель находиться в точке с координатами (100,50), то процедура LineRel(250,100) переместит его в точку с координатами (350,150) и при этом между двумя точками будет вычерчена линия. Процедура LineRel, так же как и процедура LineTo(см. выше) хорошо подходит для вычерчивания различных многоугольников или ломаных линий.

Пример 18 программы использования процедуры LineRel:

```
program primer;
  uses crt,graph;
            var i,j:integer;
             gd,gm:integer;
begin
         clrscr;
            gd:=detect;
            initgraph(gd,gm,'');
        moveto(50,50);
for i:=1 to 10 do
       begin
```

```
    if   i mod 2=0  then  j :=-150 else j :=200;
                        LineRel(i+40,j);
       end;
              readln;
          closegraph;
end.
```

В приведённом примере на экране изображается ломаная линия. Первой здесь вызывается процедура MoveTo, которая перемещает указатель текущей позиции в заданную точку экрана. Затем организуется цикл, в теле которого вызывается процедура LineRel, вычерчивающая на экране очередной участок ломаной линии. В конце программы осуществляется задержка (чтобы можно было увидеть результат - ReadLn) и выход из графического режима(closegraph). В какую сторону направить очередной участок - вверх или вниз - определяется с помощью условного оператора If. Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

Turbo Pascal позволяет вычерчивать линии различного стиля: тонкие, широкие, штриховые, пунктирные и т.д. Для того чтобы установить стиль вычерчиваемой линии необходимо воспользоваться процедурой SetLineStyle.

Заголовок процедуры:

Procedure SetLineStyle (Type, Pattern, Thick: Word)

Эта процедура задаёт новый стиль линии. Параметр Type - устанавливает тип линии, Pattern - шаблон линии, Thick - толщина линии. Тип линии задаётся с помощью набора определённых в модуле Graph констант. Перечень этих констант, соответствующие им числовые коды и образцы линий представлены в таблице 8.

Таблица 8- Константы, коды и образцы типов линий

Имя константы	код	образец
SolidLn	0	Сплошная линия
DottedLn	1	Линия из точек
CenterLn	2	Штрихпунктирная линия
DashedLn	3	Пунктирная линия
UserBitLn	4	Узор линии определяет пользователь

Что же касается параметра Thick, то он может принимать только два значения представленные в таблице 9.

Таблица 9- Константы и коды толщины линии

Имя константы	код	комментарий
NormWidth	1	Толщина линии 1 пиксель
ThickWidth	3	Толщина линии 3 пикселя

Стиль линии, определённый с помощью процедуры SetLineStyle, в дальнейшем используется различными процедурами, отображающими на экране разного рода линии, дуги и фигуры.

Пример 19: программы использования процедуры SetLineStyle:

```
program primer;
      uses crt,graph;
        var i,x,y:integer;
            gd,gm:integer;
begin
           clrscr;
             gd:=detect;   Initgraph (gd,gm,");
         randomize;
           SetBkColor(15); SetColor(random(14));
              SetLineStyle(0,0,l);
Line(50,50,400,50);
          SetColor(random(14)); SetLineStyle(1,0,1);
Line(50,75,400,75);
          SetColor(random(14));  SetLineStyle(2,0,l);
Line(50,100,400,100);
         SetColor(random(14)); SetLineStyle(3,0,l);
Line(50,125,400,125);
         SetColor(random(14));  SetLineStyle(0,0,3);
Line(50,150,400,150);
          SetColor(random(14));SetLineStyle(1,0,3);
Line(50,175,400,175);
         SetColor(random(14)); SetLineStyle(2,0,3);
Line(50,200,400,200);
        SetColor(random(14));SetLineStyle(3,0,3);
Line(50,225,400,225);
                readln;
         closegraph;
end.
```

В приведённом примере на экране изображаются горизонтальные линии. Первой здесь вызывается процедура SetBkColor, которая устанавливает цвет фона, в данном случае белый. Затем случайным образом выбирается цвет вычерчиваемой линии - процедура SetColor. После этого устанавливается стиль линии - процедура SetLineStyle. В конце программы осуществляется задержка (чтобы можно было увидеть результат - ReadLn) и выход из графического режима(closegraph). В данном примере показаны все типы линий. Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы

9.3.7. Незаполненные и заполненные фигуры

Без построения различных графических фигур не обходиться ни одна серьёзная графическая система. Например, при построении меню используются прямоугольники, в которые помещаются названия режимов; круговые диаграммы строятся из окружностей и секторов и т.д. Библиотека Graph содержит ряд процедур, которые формируют различные фигуры на основе задаваемых параметров. Цвет, стиль и толщина линии для вычерчивания берутся по умолчанию или устанавливаются соответственно процедурами SetColor и SetFillStyle.

1.Для построения прямоугольных фигур применяется процедура Rectangle.

Заголовок процедуры:

Procedure Rectangle (x1, y1, x2, y2: integer)

Данная процедура отображает прямоугольник с использованием текущего стиля и цвета линии. Параметры (x1,y1) и (x2,y2) - координаты соответственно верхнего левого и нижнего правого углов прямоугольника.

2. Процедура *Bar* рисует прямоугольник, область внутри его залита по текущему шаблону.

Еще более наглядное представление информации при рисовании диаграмм позволяет получить процедура

Bar3D(x1,y1,x2,y2:Integer; D3:Word;Top:Boolean)

Где параметр *Top* задает режим отображения верхней плоскости.

True – отображать, *TopOn*

False – не отображать, *TopOff*

Пример 20 программа использования процедуры Rectangle:

```
program primer;
   uses graph;
        var gd,gm: integer;
begin
```

```
        gd:=detect;
              initgraph(gd,gm,'');
       SetBkColor(15);  SetColor(red); SetLineStyle(0,0,3);
Rectangle(100,100,3 00,200);
       SetColor(green); SetLineStyle(1,0,1);
Rectangle(350,100,550,200);
       SetColor(blue);   SetLineStyle(2,0,l);
Rectangle(100,250,300,350);
      SetColor(yellow);SetLineStyle(3,0,3);
Rectangle(350,250,550,350);
      readln;
       closegraph;
end.
```

В приведённом примере на экране изображаются четыре прямоугольника. Перед рисованием каждого прямоугольника устанавливается цвет рисования (процедура SetColor) и стиль линии (процедура SetLineStyle), затем с помощью процедуры Rectangle рисуется прямоугольник заданного размера. В конце программы осуществляется задержка (чтобы можно было увидеть результат - ReadLn) и выход из графического режима(closegraph). Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

Пример 21: построение двух прямоугольников используя процедуры - *Rectangle*, *Bar,* соединенных прямыми линиями.

```
      Program lin1;
    Uses graph, crt;
    Var gd,gm:integer;
        Begin
            gd:= detect;
           Initgraph(gd,gm,'');
  Line(20,100,40,100);  Rectangle(40,60,80,120);
  Line(80,100,140,100);  Bar(140,80,200,120);
     Readln;
      Closegraph;
    End.
```

Для того чтобы нарисовать окружность применяется процедура Circle.
Заголовок процедуры:

Procedure Circle (x, y: integer; Rad: Word)

Данная процедура вычерчивает окружность. Параметры x и y-определяют координаты центра окружности, Rad - радиус окружности (в пикселях, по горизонтали). Например, процедура Circle(150,200,100) нарисует окружность с цент-

ром в точке с координатами (150,200) и радиусом равным 100. Окружность вычерчивается текущим цветом, установленным с помощью процедуры SetColor. При этом используется стиль линии, определённый процедурой SetLineStyle с единственным исключением: каким бы ни был текущий тип линии (точечная или пунктирная), окружность всегда рисуется сплошной линией.

Пример 22: программа использования процедуры Circle:

```
program okrugnost;
    uses graph;
       var gd,gm:integer;
begin
        gd:=detect;
initgraph(gd,gm,");
       SetBkColor(yellow); SetColor(red); SetLineStyle(0,0,3);
Circle(100,100,50);
      SetColor(green);   SetLineStyle(0,0,3);
Circle(400,l 00,75);
      SetColor(blue);  SetLineStyle(0,0,l);
Circle(250,300,90);
           readln;
        closegraph;
end.
```

В приведённом примере на экране на жёлтом фоне изображаются три окружности разного радиуса, цвета и толщины линии. Цвет устанавливает процедура SetColor, а толщину линии процедура SetLineStyle. В конце программы осуществляется задержка (чтобы можно было увидеть результат - ReadLn) и выход из графического режима(closegraph). Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

Пример 23: использования процедуры Circle:

```
program krugi2;
      uses graph;
          var gd,gm: integer;
               x,y:real;
begin
          randomize;
   gd:=detect; initgraph(gd,gm,");
      while x<=200 do
begin
    setcolor(random(15));
```

```
    circle(random(600),random(400),random(50));
      x:=x+1;
       Delay(700);
end;
         readln;
        Closegraph;
end.
```

В этой программе на экран выводятся 200 окружностей разного радиуса и цвета. Цвет окружности выбирается случайным образом с помощью процедуры SetColor. Также случайным образом определяется координаты центра окружности и радиус. Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

Пример 24: Построение четырех внутренних с радиусом 10 и четырех внешних окружностей с радиусом 20 .

```
    Program krug;
Uses graph, crt;
Var Gd,gm,I,x:integer;
Begin
Gd:= detect; Initgraph(gd,gm,'  ');
 x=x+60;
For I:=1 to 4 do
begin
circle (x,80,20); circle(x,80,10); x:=x+40;
end;
       readln;
             closegraph;
 end.
```

Для имитации объемных фигур, используются дуги. Они вычерчиваются с помощью процедуры:

Arc(x,y:Integer; StAngle, EndAngle, Radius:Word),

где *x,y* – центр окружности;
StAngle – начальный угол;
EndAngle – конечный угол, отсчитывается от горизонтальной оси против часовой стрелки.
Radius – радиус.
Если *StAngle=0 EndAngle=359*, то вычерчивается полная окружность.
Построение круговой диаграммы показано на рисунке 3.

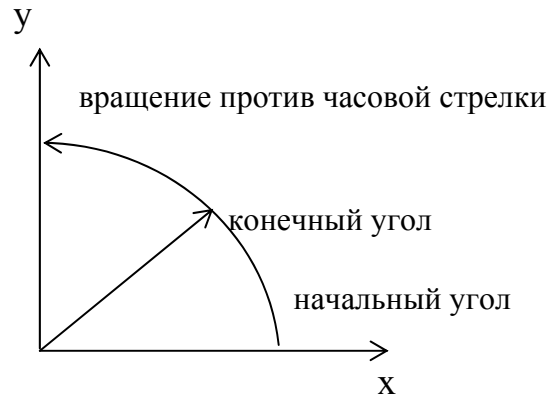


рисунок 3- Круговая диаграмма

Процедура *Arc(x,y:Integer; StAngle, EndAngle, Radius:Word)*, рисует дугу окружности: углы отчитываются против часовой стрелки и указываются в градусах. Нулевой угол соответствует горизонтальному направлению вектора слева направо. Если задать значение начального угла 0 и конечного 359, то будет выведена полная окружность.

Пример 25. Построение дуги через процедуру Arc(*x,y, StAngle, EndAngle, Radius*).

```
Program arg1;
        Uses graph, crt;
  Var Gd,Gm, Integer;
    Begin
   Gd:=detect;
   Initgraph(Gd,Gm, ‘  ’);
      Arc (80,100,45.180,30);
       Readln;
   Closegraph;
   End.
```

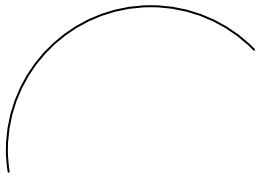


Рисунок 4- построения дуги с координатами центра 80,100

Вот как выглядят две дуги: одна с углами 0 и 100, вторая 300 и 540 (рисунок 5)

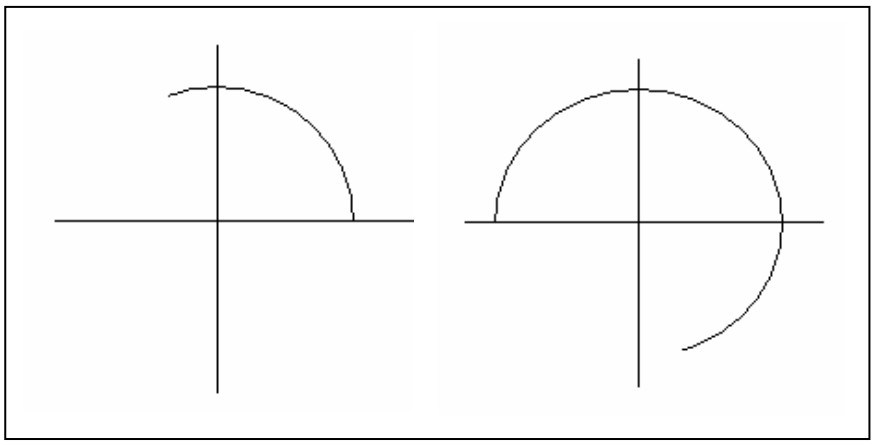


Рисунок 5- Пример дуг с разными начальными и конечными углами

Пример 26: программа использования процедуры Arc:

```
program primer;
                uses graph;
                  var gd,gm,i:integer;
begin
             gd:=detect;   initgraph(gd,gm,'');
   SetBkColor(15);
                    Randomize;
                      SetLineStyle(0,0,1);
          For i:=1 to 20 do
 Begin  SetColor(random(15));Arc(300,200,90+i*10,350+i*10,50+i*5);  end;
           readln;
         closegraph;
end.
```

В этой программе на экран выводятся 20 дуг окружностей разного радиуса и цвета. В начале программы после инициализации графики(initgraph) устанавливаем цвет фона с помощью процедуры SetBkColor. Затем с помощью процедуры SetLineStyle устанавливаем стиль линии, которыми будут вычерчиваться дуги. После этих установок с помощью цикла рисуются 20 дуг, у которых одинаковый центр (точка с координатами 300 и 200) но разные начальные и конечные углы, а также разные радиусы. Цвет дуги выбирается случайным образом с помощью процедуры SetColor. Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

Если существует необходимость нарисовать эллипс, то применяется процедура Ellipse.

Заголовок процедуры:

Procedure Ellipse (x,y:integer; BegA, EndA, RX, RY:Word);

Эта процедура вычерчивает эллипсную дугу. Параметры x и y – координаты центра эллипса, BegA и EndA – соответственно начальный и конечный углы дуги, RX и RY – горизонтальный и вертикальный радиусы эллипса в пикселях (рисунок 6).

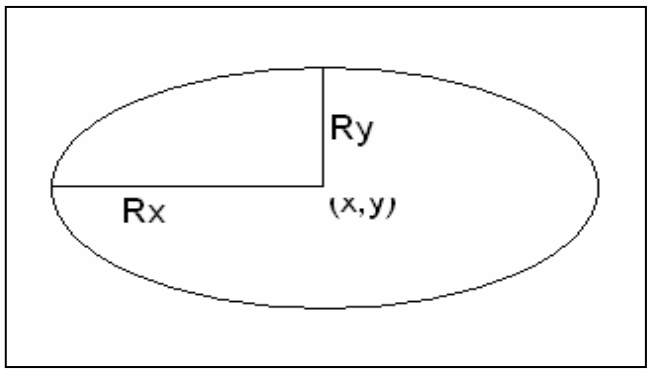


Рисунок 6- Пример эллипса

Эллипс вычерчивается текущим цветом, установленным с помощью процедуры SetColor. При этом используется стиль линии, определённый процедурой SetLineStyle с единственным исключением: каким бы ни был текущий тип линии (точечная или пунктирная), эллипс всегда рисуется сплошной линией.

Пример 27: программа использования процедуры Ellipse:

```
          …….
uses graph;
        var gd,gm,i:integer;
begin
             gd:=detect;
                    initgraph(gd,gm,'');
    SetBkColor(7); SetLineStyle(0,0,1);
      SetColor(4);
Ellipse(100,150,0,360,30,80);
     SetColor(1);
Ellipse(300,150,0,180,30,80);
       SetLineStyle(0,0,3);
      SetColor(8);
Ellipse(200,300,90,360,30,80);
      SetLineStyle(0,0,1)
       ;SetColor(13);
Ellipse(500,150,90,270,30,80);
      SetLineStyle(0,0,3);
         SetColor(2);
Ellipse(400,300,30,340,30,80);
      readln;
      closegraph;
end.
```

В этой программе на экран выводятся 5 дуг эллипсов. Каждая дуга эллипса имеет свой центр свои начальный и конечный радиусы и начальные и конечные углы. Перед вычерчиванием каждой дуги эллипса устанавливается стиль линии (процедура SetLineStyle) и цвет (процедура SetColor). Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

Помимо незаполненных фигур, средства модуля Graph позволяют выводить на экран также заполненные фигуры. С использованием заполнений мы познакомились в разделе «Цвета, палитры, заполнения» (см. выше). Далее мы рассмотрим процедуры, которые выводят на экран уже заполненные фигуры.

Для того, чтобы заполнить замкнутую область экрана, необходимо применить процедуру FloodFill.

Заголовок процедуры:

Procedure FloodFill(x,y:integer; Border:word);

Данная процедура заполняет произвольную замкнутую область используя

стиль заполнения (шаблон и цвет).

Параметры x и y – координаты произвольной точки в области, которую требуется заполнить, Border – код цвета линии, ограничивающий заполняемую область. Нужный стиль заполнения должен быть предварительно задан процедурой SetFillStyle. Если точка с координатами (x,y) находится внутри замкнутой области, заполненной окажется эта область. Если точка находится снаружи замкнутой области, заполняется все пространство вне области. Если же область окажется незамкнута, заполненным окажется весь экран.

Пример 28 программа использования процедуры FloodFill:

```
program primer;
uses graph;
        var gd,gm,i:integer;
begin
        gd:=detect;
        initgraph(gd,gm,'');
SetBkColor(7);  SetLineStyle(0,0,3);SetFillStyle(8,5);
SetColor(4);
         Circle(GetMaxX div 2,GetMaxY div 2,200);
SetColor(1);
       Line(320,45,490,333);
          Line(147,333,490,333);
              Line(147,333,320,45);
SetColor(2);
            Rectangle(250,180,390,290);
FloodFill(GetMaxX div 2,GetMaxY div 2,2);
        readln;
   closegraph;
end.
```

В приведённом примере показано как применяется процедура FloodFill. Прежде чем закрашивать замкнутую область нарисуем следующий рисунок. Сначала установим цвет фона с помощью процедуры SetBkColor. Потом с помощью процедуры SetLineStyle установим стиль линии, которой будет вычерчиваться рисунок. И, наконец, процедура SetFillStyle позволит нам определить шаблон и цвет заполнения замкнутой области. После этого установив цвет рисования (процедура SetColor) нарисуем окружность красным цветом. Внутри окружности нарисуем линиями треугольник синим цветом, а внутри треугольника нарисуем прямоугольник синим цветом (см. рис 2.5а).

Весь рисунок мы сделаем таким образом, чтобы центр окружности попал в треугольник и в прямоугольник. После построения такого рисунка применим процедуру FloodFill. В качестве первых двух параметров этой процедуры возьмём координаты центра окружности. Третий параметр процедуры FloodFill мы будем менять. Если в качестве третьего параметра мы возьмём число 2 (что соответствует цвету нарисованного прямоугольника) то на нашем рисунке

закраситься только прямоугольник (см. рисунок 7б). Если вместо числа два мы поставим число 1 (что соответствует цвету нарисованного треугольника) то закраситься треугольник и всё что находиться внутри него (см. рисунок 7в). Ну и, наконец, если в качестве третьего параметра процедуры FloodFill мы возьмём число 4 (что соответствует цвету нарисованной окружности) то произойдет закраска этой окружности и всего того, что находиться внутри неё (см. рисунок 7г).

Запустите интегрированную среду программирования, введите текст программы и запишите на диск, затем откомпилируйте его и проверьте действие программы.

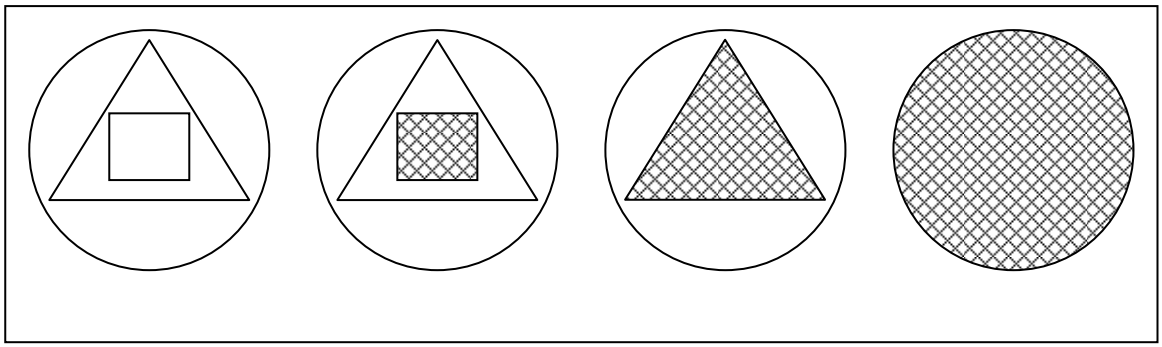


Рисунок 7 - Пример работы процедуры FloodFill

Пример 29: Построение 5 внутренних эллипсов

```
   Program ellips;
Uses graph, crt;
        Var  gd,gm,x,I :integer;
Begin
       Gd:= detect;   Initgraph(gd,gm,'');
            x:=0;
For I:=1 to 5 do
   Begin
      Ellipse(180,200,0,360,50,89+x);    x:=x+10;
    end;
         readln;
     closegraph;
end.
```

Закрашенный многоугольник

FillPoly(число вершин,двумерный массив координат вершин);

В строке массива записывают координаты X, У одной вершины.

Эта процедура по своему действию очень похожа на процедуру *DrawPoly*.

Фрагмент программы рисования закрашенного треугольника :

```
  ……….
Const T:array[1..3,1.2] of Integer= ((20,50),(10,30),(50,10));
```

```
Begin
    ……..
SetFillStyle(1,2); {1- Сплошная закраска  зеленым цветом- 2}
SetPoly(3,T); {Рисуется закрашенный треугольник}
     ….……
```

Процедуры, которые используются для заполнения секторов окружностей и эллипсов. Рисование сектора эллипса, который будет залит цветом по текущему шаблону.

Sector (X,Y:Intejer; StAngle, EndAngle,XR,YR:Word) параметры этой процедуры аналогичны *Arc,Ellipse.*

Чтобы разделить окружность на несколько секторов
PieSlice (X,Y: Intejer; StAngle, EndAngle, Radius: Word) вычеркивает и заполняет текущим узором сектора круга
Где *X,Y* – центр окружности.

Высота сектора задается значением 2 углов *StAngle, EndAngle*.

1.*FillEllipse (X,Y:Integer; XR,YR:Word)* Процедура служит для вычеркивания контура эллипса текущим цветом и заполнение ограниченной контурами области экрана текущим узором.

Если чередовать операторы *SetFillStyle* и *PieSlice* можно изобразить круговую диаграмму.

2.Заполнение более сложной фигуры

Floodfill – заполнение области расположенной в нутрии замкнутого контура, либо вне его.

Floodfill(X,Y:Integer;Border:Word)

X,Y – координаты в нутрии или вне замкнутого контура.

Border – задает цвет контура.

Примеры 30; рисования окружностей и их закрашивание .

```
Program krug2;
Uses graph, crt;
Var Gd,gm,i :integer;
Begin
gd:= detect;
        Initgraph(gd,gm,'   ');
      i:=0
Repeat
    Setcolor(15); circle (320,235,230); setfillstyle(1,i); floodfill(320,235,15);
           readln;   i:=i+1;
until i>15;
      closegraph;
end.
```

Примеры 31:

```
 Program krug3;
Uses graph, crt;
Var gd,gm :integer;
Begin
        gd:= detect;
        Initgraph(gd,gm,'  ');
      Ellipse(220,150,180,90,100,100);
 Line(220,150,120,150);
         Line(220,150,220,50);
              Circle(170,100,50);
      Ellipse(170,100,0,360,50,10);
                   readln;
         closegraph;
   end.
```

Пример 32:

```
 Program krug4;{ рисование эллиптических дуг}
Uses graph, crt;
Var gd,gm,x,y :integer;
Begin
        gd:= detect;
        Initgraph(gd,gm,'  ');
             setBkcolor(Blue)
        x:=getmaxx div 2;
       y:= getmaxy div 2;
  repeat
   if random(100)=1 then setcolor(random(16));
       x:=x+ random(5)-2;
       y:=y+ random(5)-2;
     ellipse(x,y,340,200,50,40);
   until keypressed;
                   readln;
         closegraph;
   end.
```

Пример 33:

```
 Program krug5; {красная окружность разделенная на несколько секторов и
сектора залиты синей клеткой}
Uses graph, crt;
Var gd,gm :integer;
Begin
        gd:= detect;
        Initgraph(gd,gm,'  ');
```

```
Setfillstyle(8, lightblue);
      Setcokolor(12;
        Setlinestyle(2,0,3);
        Circle(320,235,230);
        Pieslice(320,235,30,90,230);{деление окружности на несколько секторов}
            readln;
        closegraph;
  end.
```

9.3.8 Работа с текстом в графическом режиме

Процедуры:

OutText – вывод на экран строки текста, начиная с текущего положения CP.
OutText(TextString:String)

OutTextXY – по умолчанию CP совмещен с левым верхним левым углом поля, если текст по горизонтали и с правым верхним углом, если текст идет по вертикали.
OutTexXY(x,y,Text:String);
Где x,y – координаты точки начала вывода текста.

SetTextStyle(<шрифт>,<Направление>,<Размер шрифта>);
Где

- <шрифт> -значение 0-4
0-Матричный
1-триплекс
2- мелкий
3- прямой
4- готический
<Направление>-0 текст идет слева- направо
- 1 текст идет снизу-вверх, по вертикали символы на боку
*<Размер шрифта> - коэффициент увеличения от 1-10. Стандартный 8*8 пиксел*

Для явного указания положения текста по отношению с CP используется процедура.

SetTextJustify(<гор>,<вер>)Записывается до оператора OutText, OutTextXY.

Параметр *<гор>* позволяет смещать поле текста по горизонтали, а параметр <вер> по вертикали.

<гор>	*<вер>*
0 – текст справа от CP	*0 – текст сверху от CP*

1 – текст центрирован *1 – текст центрирован*
2 – текст слева от CP *2 – текст снизу от CP*

Установить, изменить толщину линий можно используя процедуру.

SetLineStyle(<Вид линий>, образец, толщина линий)
Где :
Параметр *<Вид линий>*:
SolidLn – 0 Сплошная линия
DottedLn – 1 Точечная линия
CenterLn – 2 Штрих пунктирная линия
DosheadLn – 3 пунктирная линия
UserBitLn – 4 Пользовательское определение линии
Параметр *<образец>* учитывается для случая *UserBitLn* в других случаях учитывается 0.
Параметр *<толщина>=1 для обычной толщины*
= 3 для тройной толщины

Пример 34: Вывод текста в графическом режиме по горизонтали и вертикали.

```
    Program text1;
Uses graph, crt;
Var gd,gm :integer;
Begin
        gd:= detect;
        Initgraph(gd,gm,'   ');
Settextstyle(0,0,1);{горизонтальный вывод}
          Outtext('Горизонтально');
 Settextstyle(0,1,2);{вертикальный вывод с большим размером символов}
          OuttextXY(getmaxX div 2,  getmaxY  div 2, 'вертикально');
             readln;
          closegraph;
   end.
```

10.Варианты индивидуальных заданий

10.1 Варианты заданий по теме : “Выбор в массиве по критериям”

1. Введите с клавиатуры в массив X пять целочисленных значений, выведите их в одну строку через запятую; получите для массива среднюю арифметическую.

2. Введите с клавиатуры пять целочисленных элементов массива X. Выведите на экран значения корней и квадратов каждого из элементов массива.

3. Создайте массив из пять фамилий и выведите их на экран столбиком, начиная с последней.

4. Создайте массив из пяти фамилий и выведите на экран те из них, которые начинаются с определенной буквы, которая вводится с клавиатуры.

5. Дан одномерный массив. Вставьте в него элемент L в позицию K.

6. Введите с клавиатуры целочисленные элементы матрицы 3x3, выведите исходную матрицу на экран. Умножьте каждый элемент матрицы на 3 и выведите результат на экран.

7. Создайте двумерный массив (20x15) целых чисел и найдите сумму всех его нечетных элементов.

8. Введите с клавиатуры целочисленные элементы матрицы 3x3 и вычислите сумму элементов каждого столбца.

9. Создайте матрицу 5x5, значение каждого элемента которой равно сумме номера строки и столбца, на пересечении которых он находится, и вычислите сумму элементов каждой строки.

10. Создайте массив из 15 целочисленных элементов и определите среди них минимальное значение.

12. Введите массив (не более 20) и определите, есть ли в нем элементы с одинаковыми значениями.

13. Напишите программу анализа значений температуры больного за сутки: определите минимальное и максимальное значение, среднюю арифметическую. Замеры температуры проводятся шесть раз и результаты вводятся с клавиатуры в массив T.

14. Дана матрица A, имеющая N строк и N столбцов. Сформируйте два одномерных массива. В один перешлите четные, а в другой — нечетные элементы матрицы. Выведите на экран все массивы.

15. Создайте двумерный массив вещественных чисел, имеющий 10 строк и 15 столбцов, выведите его на экран. Затем разделите каждый элемент массива на среднее арифметическое значение элементов строки, в которой они расположены и результат выведите на экран.

16. Создайте матрицу из 15 строк и 15 столбцов. Вычислите произведение суммы элементов главной диагонали на сумму элементов L-й строки.

17. Сожмите одномерный массив, удалив предшествующие минимальному элементу.

18.Найдите в одномерном массиве элементы, сумма которых максимальна, затем удалите все элементы, предшествующие тому элементу, индекс которого наибольший из двух.

19.Вычислите сумму элементов двумерного массива, индексы которых составляют в сумме заданное число К.

20. Создайте массив "шахматная доска".

21.Создайте одномерный массив, элементами которого являются суммы положительных элементов строк матрицы.

22.Найдите сумму элементов столбца и строки массива, на пересечении которых находится максимальный элемент.

23.Найдите сумму элементов массива ниже главной диагонали, произведение не равных нулю элементов выше главной диагонали и количество элементов в главной диагонали, попадающих в интервал [— 1 ;1].

24. Найдите сумму минимальных элементов главной и побочной диагонали.

25.Найдите длину наибольшего отрезка, соединяющего две точки с координатами, заданными таблицей F(2,N).

26.На плоскости своими координатами задано 40 точек. Найдите расстояние до самой удаленной (от начала координат) точки.

27.Задан список областных центров России. Присвойте переменной t название города с максимальным числом букв.

28.Сечение крыши имеет форму полукруга с радиусом R м. Требуется сформировать таблицу, содержащую длины опор, устанавливаемых через каждые R/5 м.

29.Таблица содержит 100 номеров выигрышных билетов. Проверьте, является ли билет с номером N выигрышным.

30.С 8 до 20 ч температура воздуха измерялась ежечасно. Известно, что в течение этого времени температура понижалась. Определите, в котором часу была впервые отмечена отрицательная температура.

31.В поликлинику поступили сведения о проживающих на обслуживаемой улице в виде таблиц, каждая из которых содержит номер дома, общее число жильцов дома, число детей, число пенсионеров. Всего поступили три такие таблицы. Требуется сформировать одну линейную таблицу, содержащую сведения о трех домах.

32.Дан список учеников класса и отметки каждого из учеников за выполнение двух контрольных работ. Требуется:

а) подсчитать число учеников, выполнивших первую работу на 5;

б) подсчитать число учеников, выполнивших хотя бы одну работу на 5;

в) подсчитать число учеников, выполнивших обе работы на 5;

г) подсчитать число учеников, выполнивших вторую работу на 4 и 5;

д) подсчитать число учеников, выполнивших обе работы на 4 и 5;

е) найти число учеников, выполнивших обе работы на 5, число учеников, выполнивших обе работы на 4, и число учеников, не выполнивших обе работы;

ж) найти число учеников, написавших хотя бы одну из двух работ на 5, и число учеников, не написавших хотя бы одну работу;

з) вывести список учеников, выполнивших первую работу на 5;

и) вывести список учеников, не выполнивших ни одной работы;

к) вывести список учеников, не выполнивших хотя бы одну работу.

33. В расписании движения поездов по станции Масловка указаны: номера поездов, пункты следования, время прибытия и отправления, направления следования (южное, северное, западное, восточное). Сколько поездов следует в каждом из направлений?

34.Даны список футбольных команд высшей лиги России и количество очков, набранных каждой командой в чемпионате России. Известно, что нет команд с равным числом очков. Какая из команд стала чемпионом? Составьте список команд, набравших более 15 очков.

35.Задана таблица названий товаров, выпускаемых заводом. Определите, повторяется ли в этой таблице название первого товара, и, если повторяется, удалите название первого товара из таблицы.

36.Задан список фамилий брокеров товарной биржи из N человек. Обменяйте местами фамилии брокеров: первого и последнего, второго и предпоследнего, третьего от начала и третьего от конца и т.д.

37. Заданы две таблицы. Одна содержит наименование услуг, выполняемых в доме быта, а другая — расценки за эти услуги. Удалите из обеих таблиц все, что предшествует услуге, цена которой P руб.

38. Слейте две линейные таблицы A и B в новую таблицу C, поставив элементы таблицы A на нечетные места, а элементы таблицы B - на четные.

39. В линейном массиве найти максимальный элемент. Вставьте порядковый номер максимального элемента за ним, передвинув все оставшиеся на одну позицию вправо.

40. В линейном массиве найдите индексы тех двух элементов, сумма которых максимальна среди сумм всевозможных пар. Удалите все элементы массива, предшествующие тому элементу, индекс которого наибольший из двух найденных. Преобразованный массив выведите на экран.

41. В квадратной таблице обменяйте местами элементы строки и столбца, на пересечении которых находится min из положительных элементов.

42. Наименьший элемент каждой строки прямоугольной таблицы, начиная со второй, замените наибольшим элементом предшествующей строки.

43. Даны две таблицы из N слов различной длины. Упорядочите слова по возрастанию их длин.

44. Заданы стоимость различных деталей, выпускаемых мастерской, и их названия. Отсортируйте их по стоимости и по алфавиту.

45. Задана таблица из N чисел. Сколько треугольников можно составить из этих чисел? Найдите треугольник с максимальной площадью.

46. Даны список футбольных команд высшей лиги России и количество очков, набранных каждой командой в чемпионате России. Известно, что нет команд с равным числом очков, а две команды, набравшие наименьшее число очков, покинут

высшую лигу. Какие это команды?

47. Требуется раскодировать зашифрованный текст, содержащий 150 букв. Правило шифрования известно: вместо "правильной" буквы была указана буква, стоящая в алфавите через две буквы после "правильной".

48. Ведомость на зарплату представлена как две таблицы. Одна содержит фамилии работников цеха, а вторая — их зарплату за текущий месяц.

Найдите фамилию работника, зарплата которого наименее отклоняется от средней зарплаты всех работников за текущий месяц.

Найдите фамилии двух работников с наибольшей зарплатой

Удалите из ведомости на зарплату сведения о работнике, зарплата которого минимальна.

49. Заданы три таблицы: две линейные таблицы A и B и прямоугольная таблица C. В линейной таблице A содержатся фамилии учеников 11 класса, в прямоугольной — их оценки по 10 предметам за первое полугодие, в линейной таблице B — названия этих предметов.

Составьте список неуспевающих учеников. Удалите из таблицы C все оценки ученика, имеющего больше трех двоек, а из списка A его фамилию.

Считая список учеников упорядоченным по алфавиту, внести в него строку:

Иванов Николай 5 4 3 5 4 2 4 5 3 5 Определите по какому предмету самый высокий средний балл. Найдите любых трех учеников с наибольшим средним баллом по всем предметам. Составьте список учеников в порядке убывания их среднего балла.

50. Создайте двумерный массив целых чисел. Удалите из него строку и столбец, на пересечении которых расположен минимальный элемент.

51.Дан целочисленный одномерный массив A из 10 элементов. Найдите наименьшее число K элементов, которые нужно исключить из последовательности A[1], A[2],...,A[10], чтобы осталась возрастающая последовательность.

52. Дан одномерный массив целых чисел. Найдите, сколько раз в нем повторяется самое частое число.

53. Дан одномерный массив из 10 целых чисел. Подсчитайте количество различных чисел в нем.

54. Дан одномерный массив из 10 целых чисел. Подсчитайте наибольшее число одинаковых идущих подряд в нем чисел.

55. Сформируйте одномерный массив вещественных чисел, элементы которого являются расстояниями, пройденными телом при свободном падении на землю за O, 1, 2,..., 10c.

56. В доме проживают 70 семей. Найдите номер квартиры, в которой проживает самая многочисленная семья.

57. Дан одномерный массив целых чисел. Проверьте, является ли он упорядоченным по убыванию.

58. Дан двумерный массив, содержащий фамилии учащихся и номера их телефонов. По фамилии учащегося найдите номер его телефона.

59. Дана матрица целых чисел размером 10 x!2. Напечатайте индексы всех ее седловых точек. (Седловой точкой называется элемент, который является наименьшим в своей строке и наибольшим в своем столбце или, наоборот, наибольшим в своей строке и наименьшим в своем столбце.)

60. Даны две матрицы. Получите их произведение.

61. Сформируйте и выведите на экран по строкам трехмерный массив целых чисел размером 5x5x5. Найдите максимальный элемент и укажите его положение в массиве.

62. Составьте программу, проверяющую, можно ли, меняя местами элементы одномерного массива A, получить одномерный массив B.

63. Дана квадратная таблица размером N*N. Составьте программу формирования последовательности B, элементами которой являются элементы таблицы A, расположенные под главной диагональю. В полученной последовательности найдите три наименьших элемента. Если количество элементов между первым и вторым элементом совпадает с количеством элементов между вторым и третьим, то поменяйте их местами, сохранив порядок следования (без использования дополнительного массива), преобразованный линейный массив выведите на экран в строку.

64. Составьте программу, проверяющую, образуют ли элементы двумерного массива магический квадрат (в магическом квадрате суммы чисел по всем вертикалям, всем горизонталям и двум диагоналям одинаковы).

65. Составьте программу, отыскивающую среди пар элементов одномерного массива M те, разность между элементами которых есть величина, наибольшая для данного массива.

66. Составьте программу, меняющую местами элементы матрицы симметрично побочной диагонали.

67. Составьте программу упорядочения по возрастанию элементов каждой строки двумерного массива C.

68. Составьте программу циклической перестановки столбцов двумерного массива K, при которой i-й столбец становится i+1-м, а последний столбец становится первым.

69. Составьте программу вычисления суммы тех положительных элементов двумерного массива A, которые стоят в строках, не содержащих нулевых элементов.

70. Дан числовой массив Q из N элементов. Составьте программу вычисления методом деления отрезка [a,Ь] пополам корня уравнения Sinx*Sin3x = 0,5 с заданной точностью E. В качестве A и B выбрать пару рядом стоящих элементов массива Q. Если таких пар несколько, то выберите ту из них, которая обеспечивает решение уравнения за минимальное число шагов. Найденный корень включите в массив между элементами пары, которая использовалась для нахождения корня. В преобразованной последовательности элементы, стоящие до и после включенного корня, поменяйте местами, сохранив порядок их следования и оставив корень в качестве разделительного элемента. Замену произведите без дополнительного массива. Преобразованную последовательность выведите на экран в строку.

10.2 Варианты заданий по теме : “процедуры и функции модуля CRT”.

1. Заполните экран 1600 случайными числами в диапазоне 0...9 стандартным белым цветом на красном фоне ярко-зелеными буквами. Установите зеленый фон и выведите еще 1600 случайных чисел желтым цветом. Не забудьте подключить модуль CRT, так как вам понадобятся экранные процедуры.

2. Покажите восемь возможных фонов экрана задержкой в три с. В левый верхний угол экрана выведите номер текущего цвета.

3. Выведите на экран 16 случайных чисел 16-ю разными цветами на черный экран.

4. Выведите на предварительно очищенный экран строку 'ЯРКИЙ И ТУСКЛЫЙ таким образом, чтобы первое слово было ярким, 'и' — менее ярким, а последнее слово— тусклым.

5. Выведите на предварительно очищенный экран строку 'Для продолжения работы нажмите клавишу Enter[1] таким образом, чтобы последнее слово ярко выделялось по сравнению с другими словами предложения.

6. Выведите 200 символов '*' случайного цвета в точки экрана со случайными координатами. Фон экрана-черный.

7. Постройте окно цвета LightRed и выведите в него две строки БАЗА ДАННЫХ[1] и 'СРЕДНЕЙ ШКОЛЫ' черного цвета. Сделав паузу в три секунды, очистите экран черного цвета. Обратите внимание, что дня перехода от окна к полному экрану надо обязательно выполнить оператор Window (80,25). Эта программа — подход к выводу данных в различные места экрана и первый шаг к созданию заставок, содержащих название и назначение программы.

8. Постройте два окна по верхним углам экрана и выведите в каждое сообщение 'Окно 1, "Окно 2'. Сделайте паузу три секунды и очистите экран

9. У левого края экрана создайте окно цвета lightGreen и переместите его к-правому краю экрана, имитируя движение с шагом в одну позицию.

10. Постройте "взрывающееся" (т. е. постепенно увеличивающееся от минимального к максимально возмощному размеру экрана) окно цветом Magerta для X-координат=3, для Y-координат = 1.

11. Постройте "сходящееся" (т.е. постепенно уменьшающееся от максимально возможного к строке толщиной 1 символ) окно цветом Magerta

12. Постройте два окна (цветами Blue и Green) и подпишите их 'ЧЕТНЫЕ', НЕЧЕТНЫЕ[1] желтым цветом. С клавиатуры в цикле в строке 25 вводите цифры от 1 до 9. Если число четное, выводите его цветом LightRed в окно с надписью 'ЧЕТНЫЕ' и т.д.

13. Напишите программу, в которой вывод символов строки сопровождается звуковым сигналом. Формирование сигнала оформите отдельной процедурой Beep.

14. Напишите программу, которая вводит в цикле целые числа и если число превышает 100, подает звуковой сигнал с помощью пользовательской процедуры Beep. Выход из цикла — 99.

15. Напишите программу, которая генерирует 100 целых случайных чисел в диапазоне 1...300. Если случайное число попало в диапазон 60... 100, подайте звуковой сигнал с помощью пользовательской процедуры Beep и выведите это число на экран.

16. Исследуйте датчик случайных чисел системы Турбо Паскаль. Для этого сгенерируйте 5000 случайных чисел и подсчитайте число четных и нечетных чисел. Если число четных чисел превысит 2500, подайте один звуковой сигнал, если количество нечетных чисел превысит 2500, подайте три сигнала.

17. Разработайте программу-модель датчика-анализатора температуры. В качестве источника температуры возьмите случайное число, полученное по Random(300), 100 раз. Выполните анализ диапазонов 0...100, 101...200, 201...300. В первом случае выдайте сообщение Норма[1], во втором случае — Превышение нормы[1], в третьем — сообщение Тревога[1] и один звуковой сигнал. (По принципу работы этой программы построена работа всех пожарных, химических и прочих датчиков аварийных ситуаций).

18. На основе функции Random постройте программу-исполнителя "случайной" электронной музыки. Продолжительность сигнала также случайна и может длиться от 0 до 500 мс.

19. Напишите программу, которая три раза проигрывает гамму от ДО до СИ, используя ноты ДО, РЕ, МИ, ФА, СОЛЬ, ЛЯ, СИ с частотами соответственно 262,294,330,349,392,440,494.

20. Изучите звучание различных частот, вводя с клавиатуры их значения в цикле и генерируя соответствующий

10.3 Варианты заданий по теме **“ модуль Graph”.**

1. Установите графический режим и с помощью Random заполните экран 30000 точек случайного цвета.

2. Постройте желтую линию с координатами краев (0,0),(630,0) с помощью процедуры PutPixel.

3. Постройте рамку цвета LightCyan по краям экрана с помощью процедуры Line.

4.В верхней части экрана создайте область, ограниченную рамкой цвета LightGreen и текстом в центре: "Банк данных школы № 86 г.Воронежа". Цвет для текста — LightCyan.

5. В нижней части экрана создайте область, ограниченную рамкой цвета Light(текстом в центре: "Для продолжения программы нажмите Enter". Выводимый текст, трижды изменить цвет с LightCyan на LightRed.

6. Разметьте экран на три области: заголовочную, диалогов, подсказки, как pasi оболочка IDE Turbo Pascal 5.5/6.0. Для рамок используйте цвет Yellow. Соответствен ведите сообщения: "Заголовок", "Рабочая область", "Подсказка" цветом LightCyan.

7. Используя произвольный цвет, постройте двухмерный закрашенный

прямоуго и внизу сделайте подпись белого цвета: "1992". Параметры для SetFillStyle = (1,3).

8. Нарисуйте "бублик" желтого цвета произвольного радиуса.

9. Разметьте экран на три области: заголовочную, диалогов, подсказки. Для рамо пользуйте цвет Yellow. Соответственно выведите сообщения: "Заголовок", "Рабочш ласть", "Подсказка" цветом LightCyan. Центральную часть, исключая рамку, оформит! окно с фоном Cyan и заполните его 30000 точек случайного цвета. Для очистки и уста» фона графического окна используйте процедуру Bar.

10. Нарисуйте круг цвета произвольного радиуса, закрасив цветом Cyan все, что ле все этого круга.

11. В центре экрана нарисуйте эллипс, закрашенный цветом LightGreen, всю осталы площадь экрана сделайте фиолетовой.

12. Находясь в графическом режиме, присвойте двум переменным целые значения, с. жите их и выведите результат в центр экрана. Так как вывести число в графике нельзя, *ц* образуйте результат в строку отдельной пользовательской функцией. Используйте процедру Str.

13. Находясь в графическом режиме, присвойте двум переменным дробные значени сложите их и выведите результат в центр экрана. Так как вывести число в графике нельз преобразуйте результат в строку отдельной пользовательской функцией. Используйте про цедуру Str.

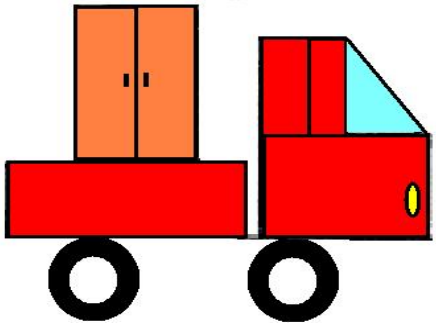
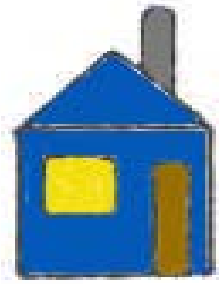
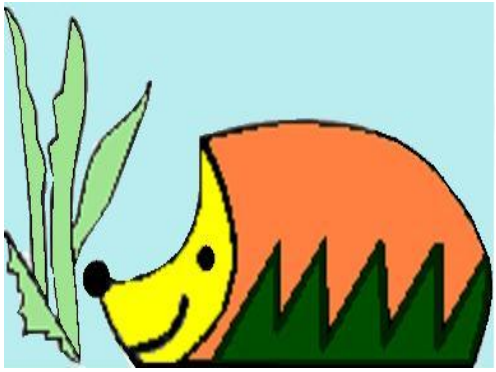
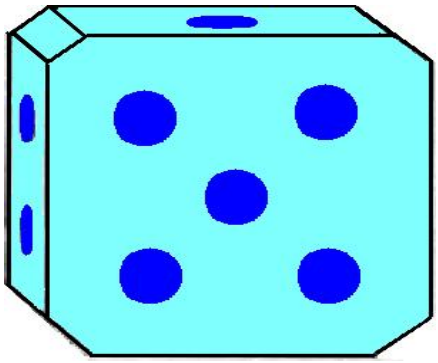
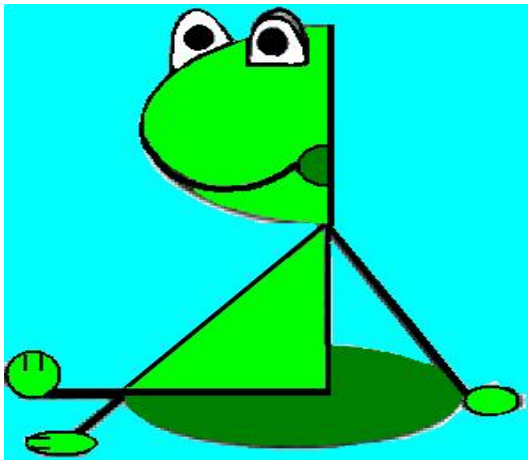
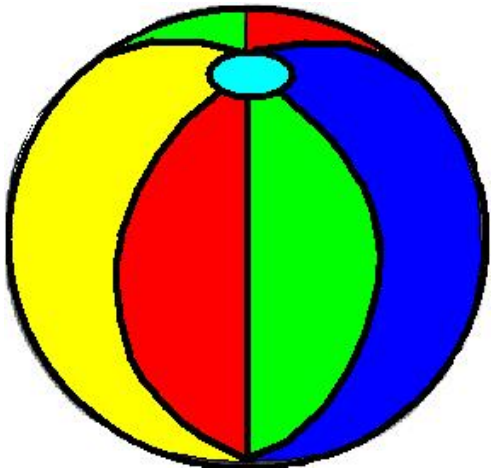
14. Постройте оси X,Y и начертите цветом LightRed график функции Y-~X до X=1..400. Учитывая то, что отсчет будет вестись от левой нижней точки экрана, функцшв X=Y будет выглядеть как Y=349—X.

15. Заданы пять значений количества заболевших гриппом в классе за последние пять лет: 12 (1990), 30 (1991), 11 (1992), 21 (1993), 4 (1994). С помощью BarЗD постройте диаграмму, отражающую динамику (в высоте столбца) заболеваемости. Под каждым столбцом укажите год. Исходные данные опишите в типизированном массиве.

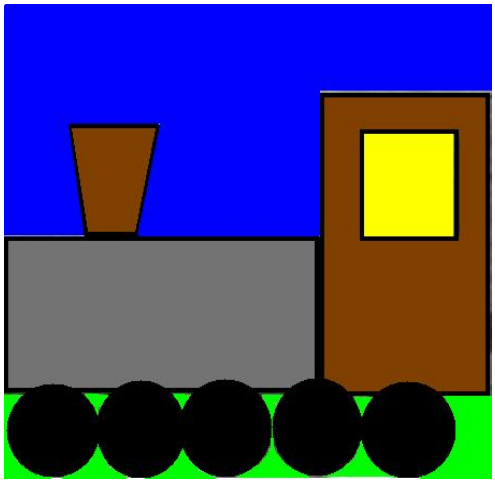
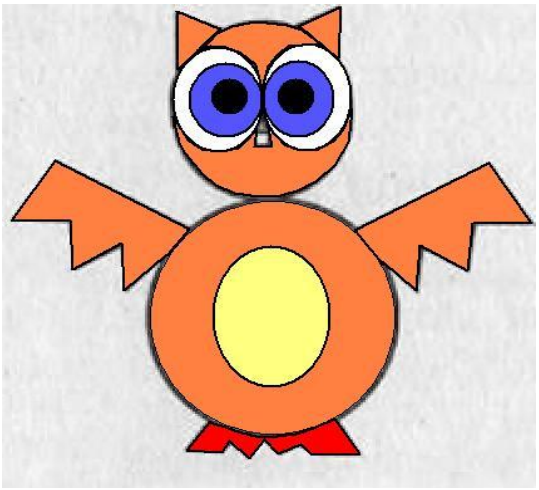
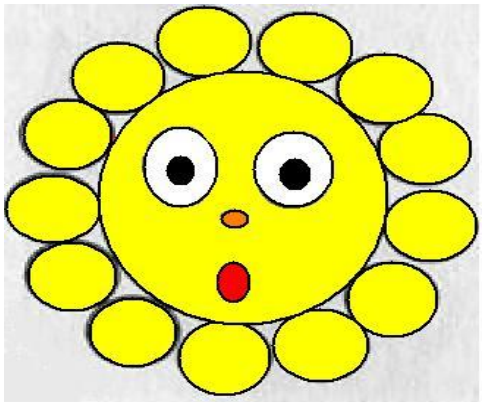
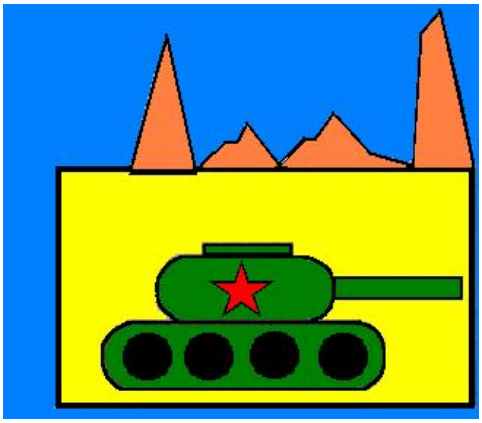
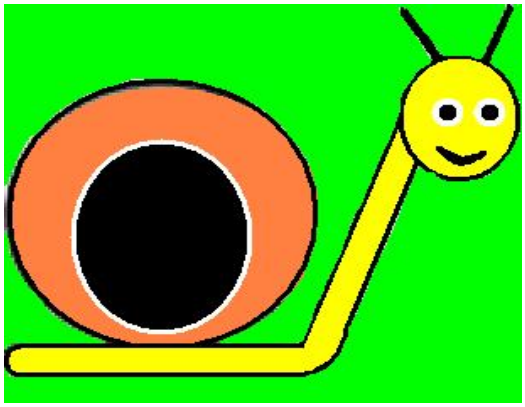
10.3.1 Эскизы индивидуальных заданий по теме **“ модуль Graph”.**

В таблице 6 представлены графические варианта индивидуальных заданий по теме **модуль Graph**

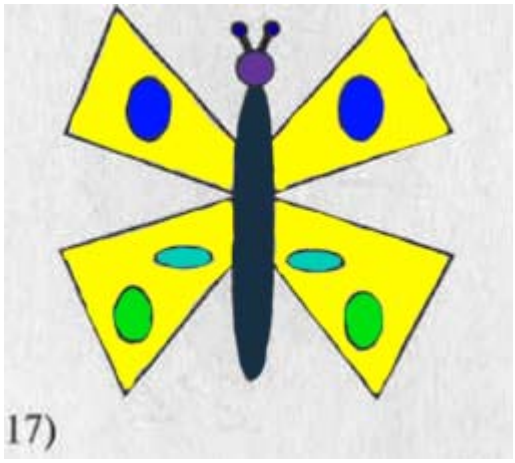
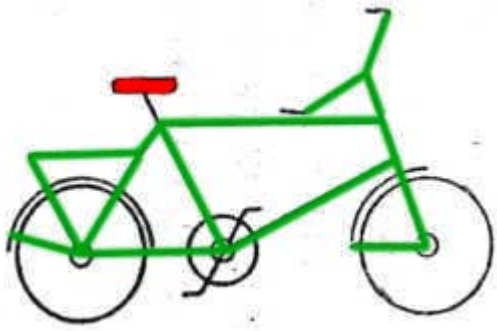
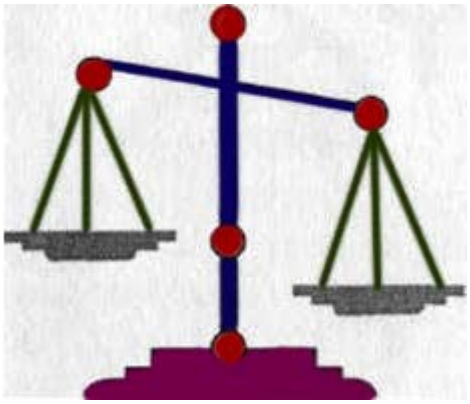
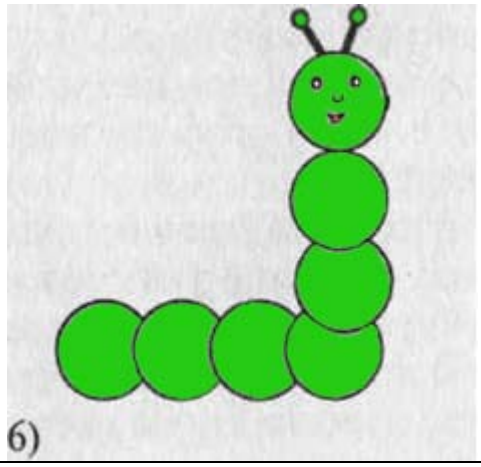
Таблица 6 - варианта индивидуальных заданий

	
1 Грузовик	2 Домик
	
3 Ежик	3 Кубик
	
5 Ллягушка	6 Мячик

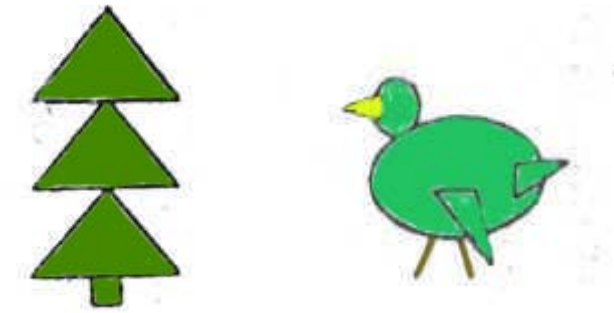
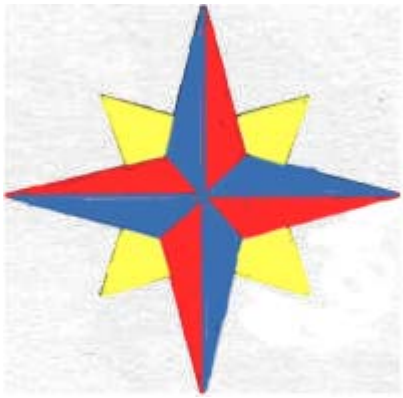
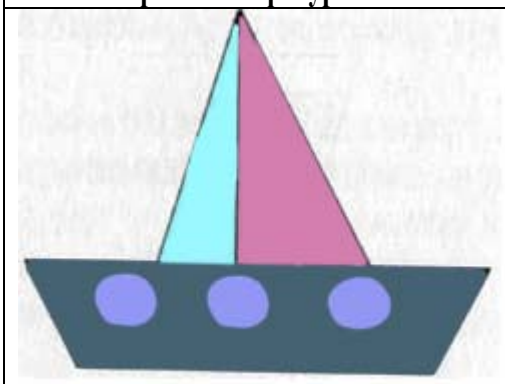
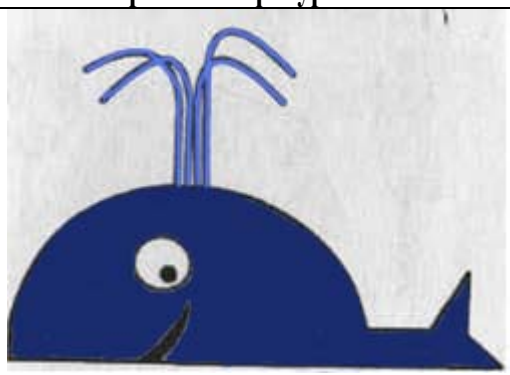
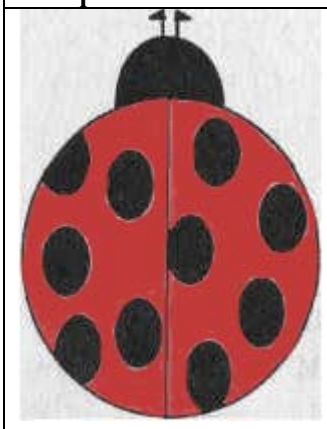
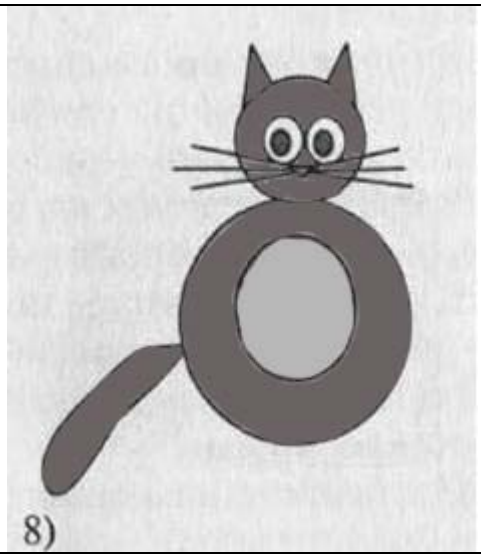
Продолжение таблицы 6

	
7 Паровозик	8 Сова
	
9 Солнышко	10 Танк
	
11 Улитка	**12 Чашка**

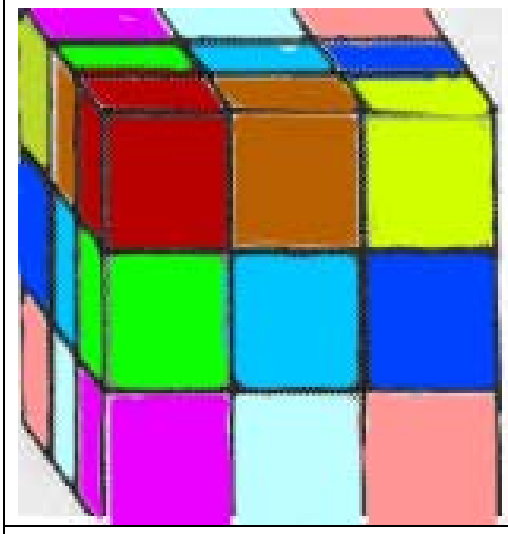
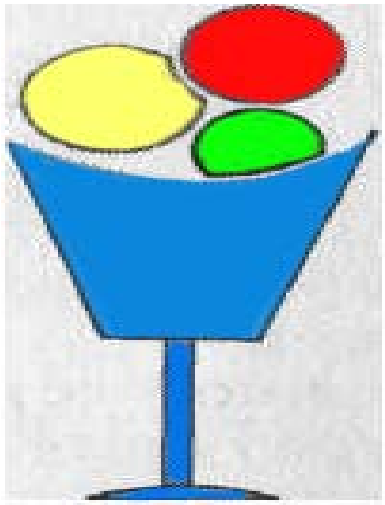
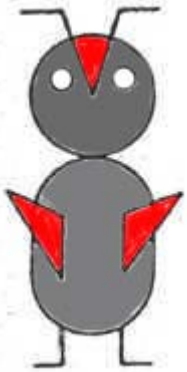
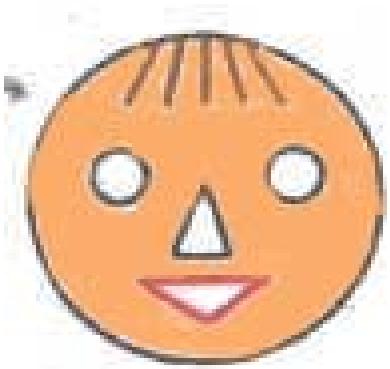
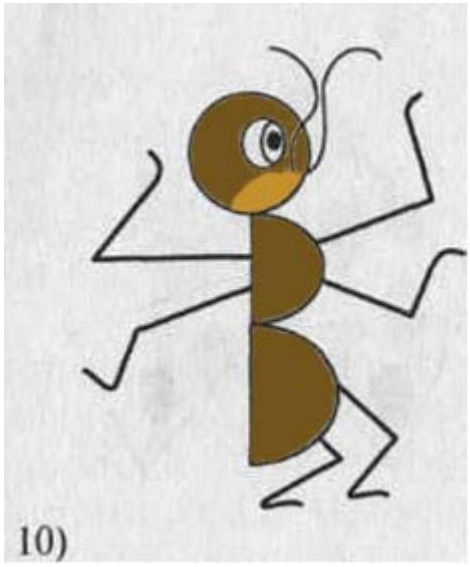
Продолжение таблицы 6

	
13 Бабочка	14 Велосипед
	
15 Весы	16 Гусеница
	
17 Песочница	18 Грибки

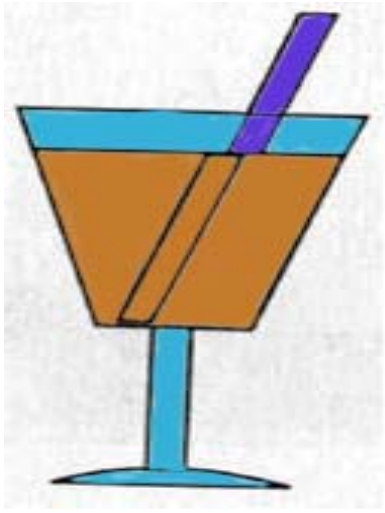
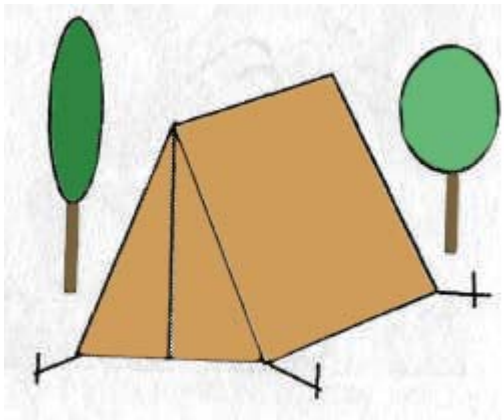
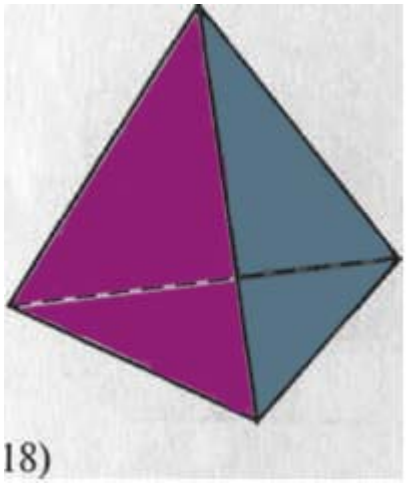
Продолжение таблицы 6

	
19 Геометрические фигурки	**20 Геометрические фигурки**
	
21Кораблик	**22 Кит**
	
23 Коровка	**24 Кошка**

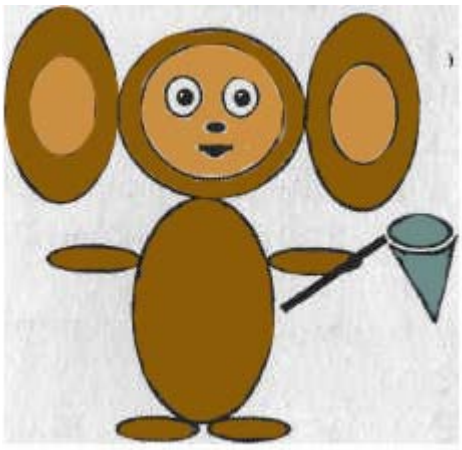
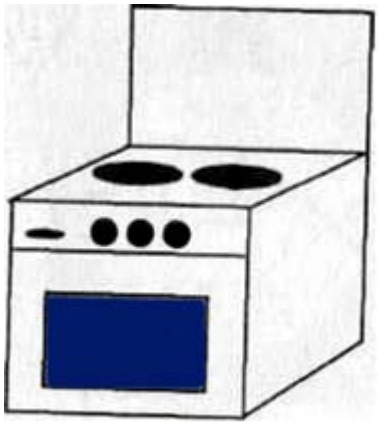
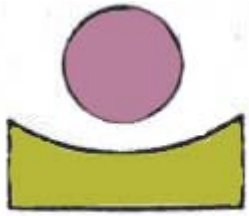
Продолжение таблицы 6

	
25 Кубик рубик	**26 Вазочка**
	
27 Муравей	**28 Геометрические фигурки**
	
29 Геометрические фигурки	**30 Муравей**

Продолжение таблицы 6

	
31 Фужер	**32 Чайник**
	
33 Часы	**34 Будильник**
	
35 Палатка	**36 Геометрические фигурки**

Продолжение таблицы 6

	
37 Чебурашка	**38 Плита**
	
39 Спортсмен	**40 Геометрические фигуры**

11 Вопросы для подготовке к зачету по практике

11.1 Контрольные вопросы по теме: Массивы

1. Что такое массив?
2. Как определить местоположение элемента в массиве?
3. Что такое индекс? Каким требованиям он должен удовлетворять?
4. Особенности расположения элементов массива в памяти ЭВМ. Особенности расположения в памяти элементов многомерных массивов.
5. Каким образом задается описание массива, что в нем указывается?
6. Общие и отличительные черты одномерных, двумерных и n-мерных массивов.
7. В каких операциях могут участвовать массивы и какие к ним при этом предъявляются требования?
8. Каким образом в Паскале задается обращение к элементу массива?
9. Почему при описании массивов предпочтительнее употреблять константы, а не указывать размеры массива в явном виде?
10. Что называют инициализацией массива, и зачем она применяется?
11. Что называется контрольной точкой останова? Как задать,ее положение? Как запустить исполнение программы до контрольной точки останова?
12. Что называется сортировкой массива? Какие методы сортировки вы знаете, опишите их существенные отличия.
13. Как задать имена переменных или выражения для просмотра их значений в окне просмотра при исполнении программы по шагам?
14. Что вы понимаете под поиском? Отличительные черты линейного и бинарного поиска? В каких случаях целесообразнее использовать линейный или бинарный поиск?
15. Что называется главной (побочной) диагональю матрицы? Что такое транспонирование матрицы?

11.2 Вопросы по теме:“Управление экраном и звуком. Компьютера”

1. Каково назначение модуля CRT? Как подключить данный модуль к программе?
2. Какой стандартной процедурой осуществляется установка текстовых режимов?
3. Общие и отличительные черты процедур ClrScr, ClrEol, DelLine и InsLine.
4. Назначение процедуры GoToXY, функций WhereX и WhereY.
5. Какова структура байта атрибута цветности для цветных текстовых режимов?
6. Каково назначение переменной TextAttr? Как влияют на ее значение процедуры TextColor и TextBackGround?
7. Как задается эффект мерцания выводимых символов?
8. Какие процедуры используются для управления яркостью?

9. Какой эффект дает использование буфера дисплея для вывода информации?

10. Какая процедура используется для организации окон в Турбо Паскале? Каково назначение специальных переменных WindMin и WindMax?

11. Перечислите назначение стандартных процедур Sound, NoSound и Delay модуля CRT?

12. Какая типичная конструкция используется при организации звуковых эффектов?

13. Каков будет результат выполнения следующей программы:

```
program -music; uses Crt;
        begin
    Sound(12300);
Delay(SOO) ;
     NoSound;
    end.
```

11.3 Контрольные вопросы по теме: “модуль Graph”

1. Чем объяснить разные возможности текстового и графического режимов видеомонитора?

2. За счет каких элементов компьютера обеспечивается аппаратная поддержка

3. Какие основные типы видеомониторов персональных компьютеров типа IBM PC вы знаете? Их отличия.

4. Что такое видеоадаптер, каково его назначение, из каких основных частей он состоит?

5. Каково назначение графических драйверов? В каких файлах они находятся? Назначение процедуры RegisterBGIDriver?

6. Назначение стандартного библиотечного модуля GRAPH? Как запустить графическую систему?

7. Как включить графический режим? Назначение процедуры InitGraph и ее параметров: DriverVar, ModeVar, PathToDriver? Как перейти в текстовый режим и обратно? Как закрыть графический режим?

8. Какие стандартные функции можно использовать для своевременного обнаружения и нейтрализации ошибок в графическом режиме?

9. Какова система координат, используемая при построении графических изображений на экране?

10. Какую роль выполняет в графическом режиме невидимый текущий указатель CP (Current Pointer)? Как его переместить?

11. Что такое окно? Как создать окно в графическом режиме? Как установить фон графического окна?

12. Какие процедуры используются для работы с графическим примитивом точкой?

13. Какие процедуры используются для работы с графическим примитивом

— линией?

14. Какие процедуры используются для вывода текста в графическом режиме? Каковы особенности вывода численных данных в графическом режиме?

15. Каково отличие растровых и векторных шрифтов? Как установить шрифт, задать его размер, ориентацию?

16. Как можно получить информацию о текущей палитре? Как задать цвет фона, рисунка? Как изменить всю палитру?

17. Какие процедуры используются при рисовании прямоугольников и многоугольников?

18. Опишите полярную систему координат. Какие стандартные процедуры используются при рисовании окружностей, дуг, эллипсов, секторов?

19. Какие процедуры используются для установки цвета и стиля геометричес ких фигур?

Список использованных источников

1 **Семагин И.Г.** Основы программирования [Текст] /И.Г.Семагин, А.П.Шестаков. –М.: Академия, 2004. –385 с.

2 **Попов В.Б.** TURBO PASCAL для школьников [Текст] /В.Б.Попов. –М.: Финансы и статистика, 2001. -518 с.

3 **Житкова О.А.** Справочные материалы по программированию на языке Паскаль [Текст] /О.А.Житкова, Е.К.Кудрявцева. –М.: Интеллект-центр, 2002. -77 с.

4 **Меженный О.А.** TURBO PASCAL. Учитесь программировать [Текст] /О.А.Меженный. –М.: Диалектика, 2001.-88 с.

5 **Фаронов В.В.** TURBO PASCAL 7.0 [Текст] /В.В.Фаронов. –М.: Нолидж, 2000. -312 с.

6 **Марченко А.И.** Программирование в среде TurboPascal 7.0 [Текст] /А.И.Марченко. –М.: Бином Универсал, 2000. – 485с.

7 **Культин Н.** Delphi Программирование на Object Pascal [Текст] /Н.Культин. –М.: BHV–Санкт-Петербург, 2002. –297с.

8.**Зубов В.С.** Программирование на языке TP [Текст] /В.С.Зубов. –М.: ТТО «Филинъ», 2000. –301с.

9. **Васюокова Н.Д.** Практикум по основам программирования язык Паскаль [Текст] /Н.Д.Васюокова, В.В.Тюляева. –М.: Высшая школа, 2000. –с.

10. **Бондарев В.М.** Основы программирования [Текст] /В.М.Бондарев, В.И.Рублинецкий, Е.Г.Качко. –М.: ТТО «Филинъ», 2000. –363с.

Приложение А

Пример оформления графика прохождения практики

График

прохождения практики для получения первичных профессиональных навыков студентов 2 курса специальности ПОВТАС

1	***Задачи практики***:	1 неделя
1.1	Стандартизация документации - отчет по практике	
1.2	*Техника безопасности и охрана труда*	
1.3	Выдача задания на практику.	
2	Модуль CRT. Установка цветом символа (изображения).	
	Процедуры действующие в окнах (Window)	
3	Составление алгоритмов и программ по теме выбор в массиве	
	по критериям.	
4	*Отладка программы выбор в массиве по критериям*	
	по индивидуальному заданию	
	Контрольный просмотр	
5	**Оформление программы индивидуального задания**	2 неделя
	с использованием процедур модуля CRT	
6	Оформление выполненных работ	
7	***Контрольный просмотр***	
8	Модуль GRAPH. Стандартные процедуры и функции модуля.	
9	Работа с текстом в модуле GRAPH. Переключение между	
	текстовым и графическим режимом	
10	Разработка эскиза и алгоритма индивидуального задания.	3 неделя
11	Разработка программы.	
12	Отладка и редактирование программы индивидуального задания	
	Составление программы с использованием процедуры	
	OutText, OutTextXY	
13	***Контрольный просмотр***	
14	***Подготовка к защите выполненных работ***	
	Защита выполненных работ	

Приложение Б

(обязательное)

Пример оформления титульного листа

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

КОЛЛЕДЖ ЭЛЕКТРОНИКИ И БИЗНЕСА

Кафедра вычислительной техники и математики

ОТЧЕТ ПО ПРАКТИКЕ

для получения первичных профессиональных навыков

КОГУ 230105.9005.02П

порядковый номер по списку

Руководитель работы
_____Дель Л. А.
«___»_____ 2008 г.

Исполнитель
Студент гр. 23П2
_____Дель В.Г.

Оренбург 2008

Приложения В
(обязательное)
Пример оформления бланка задания

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

КОЛЛЕДЖ ЭЛЕКТРОНИКИ И БИЗНЕСА

Кафедра вычислительной техники и математики

Задание на практику

По дисциплине «основы алгоритмизации и программирования»

1. Индивидуальное задание на составление программы выбор в массиве по критериям
2. Индивидуальное задание на составление программы с помощью модуля Graph

Дата выдачи задания: «___»____ 2008 г.
Руководитель:_____Дель Л. А.
Исполнитель:
Студент гр 23-П-2 _____Дель В.Г.

Оренбург 2008

Приложение Г
(справочное)
Пример оформления содержания

Содержание

Приложение Д
Справочная информация

Таблица Д.1. -Процедуры модуля Graph

Процедура	Действия
Arc	Рисует дугу от начального угла к конечном;используя (X,Y) как центр
Bar	Рисует полосу, используя текущий стиль и цвет
Bar3D	Рисует трехмерную полосу, используя текущие стиль и цвет
Circle	Рисует окружность, используя (X,Y) как центр
ClearDevice	Очищает экран и устанавливает текущий указатель (CP) начало
ClearViewPort	Очишаст окно
CloscGraph	Закрывает графическую сисгему
DetectGraph	проверяст аппаратуру и определяет, какой графический драйвер и в каком режиме используется
DrawPoly	Рисует многоугольник текущими цветом и типом линии
Ellipse	Рисует эллиптическую дугу от начального угла к конечному, используя (X,Y) как центр
FillEllipse	Рисуег заполненный злли пс, используя (X,Y) как центр и XRachus и YRadius как горизонтальные и вертикальные оси
FillPoty	Заполняет многоугольник, используя сканирование
FloodFill	Заполняет ограниченную область, используя текущий шаблон и цвет заполнения
GetArcCoords	Позволяст запросить координаты команды Arc
GetAspectRatio	Возвращает разрешение экрана, из которого может быть вычислен относительный аспект (Xasp/Yasp)
GetFill Pattern	Возвращает шаблон заполнения, установленный последним вызовом SetFil1Pattern
GetFillSettings	Позволяет запросить текущие шаблон и цвет, установленные SetFillStyle или SetFil1Pattern
Getlmage	Сохраняет битовый образ указанной части экрана в буфере
GetLineSettings	Возвращает текущие стиль, шаблон и толщину линии, установленные SetLineStyle
GetModeRange	Возвращает минимальный и максимальный графические режимы для данного драйвера
GctPalette	Возвращает *текущую* палитру и ее размер
GctTcxtSettings	BosBpamaer текущие шрифт, направление, размер и выравнивание текста, установленные SetTcxtStyle и SetTexUustiry
GetViewSettings	Позволяет запросить текущие параметры окна и отсечения
Graph Defaults	Устанавливает текущий указатель (CP) в начало и переустанавливает графическую систему

Продолжение таблицы Д.1

1	2
InitGraph	Инициализирует графическую систему и устанавливает устройство в графический режим
Line	Рисует линию от (X1.Y1) к (X2.Y2)
LineRel	Рисует линию от текущего указателя (CP) к точке, лежащей на заданном расстоянии
LincTo	Рисует линию от текущего указателя к (X,Y)
Move Rel	Передвигает текущий указатель (CP) на заданное расстояние от его текущей позиции
MovcTo	Передвигает текущий указатель (CP) в (X,Y)
OutText	Выводит текст на экран от текущего указателя
OutTextXY	Выводит текст на экран
PieSlice	Рисует и заполняет сектор, используя (X,Y) как центр и рисуя от начального угла к конечному
Puf Image	Выводил битовый образ на экран
PutPixel	Рисует точку (пиксел) в (X,Y)
Rectangle	Рисует прямоугольник, используя текущие цвет и тип линии
RestoreCRTMode	Восстанавливает видеорежим, который был до инициализации графики
Sector	Рисует и заполняет сектор эллипса
SetAcnvcPagc	Устанавливает активную страницу для графического вывода
SetAllPalette	Изменяет цвет палитры
SetAspectRatio	Изменяет значение относительного аспекта
SctBkColor	Устанавливает цвет фона
SetColor	Устанавливает основной цвет, которым будет осуществляться рисование
SetFillPattern	Выбирает шаблон заполнения, определенный пользователем
SctFillStyle	Устанавливает шаблон -заполнения и цвет
SetGraphBufSize	Позволяет изменить размер буферадля функций заполнения
SciGraphMode	Переводит систему в графический режим и очищает экран
SefLineStyle	Устанавливает текущие толщину и стиль линии
SetPatette	Изменяет один цвет палитры, указанный через ColorNum и Color
SetTextJustiry	Устанавливает выравнивание текста, используемое OutText и OutTextXY
SetTextStyte	Устанавливает текущие шрифт, стиль и размер текста
SetUserChatSize	Позволяст изменить ширину и высоту символа для штрихового шрифта
SetViewPort	Устанавливает текущее окно для графического вывода
SetVisualPage	Устанавливает номер видимой графической страницы
SetWriteMode	Устанавливает режим вывода (копирование илиXOR) для линий, рисуемых с DrawPoly, Line, LineRel, LineTo, Rectangle

Таблица Д.2 - Функции модуля Graph.

Функция	Действие
GetBkColor	Возвращает текущий фоновый цвет
GetColor	Возвращает текущий цвет
GetDefauItPalette	Возвращает аппаратную палитру в записи PalettcType
GetDriverNams	Возвращает строку с именем текущего драйвера
GetGraphMode	Возвращает текущий графический режим
SetMaxColor	Возращает максимальный цвет, который можно задать в SetColor
GetMaxMode	Возвращает номер максимального режима текущего загруженного драйвера
GctMaxX	Возвращает максимальный X (разрешение по горизонтали) для текущею графического драйвера и режима
GctMaxY	Возвращает максимальный Y (разрешение но вертикали) , текущего графического драйвера и режима
GetModeName;	Возращиет строку с именем указанного графического режима
GctPaletteSize	Возращает размер таблицы палитры
GctPixel	Воэвращаег цвет точки в (X,Y)
GelX	Возвращает координату X текущей позиции (CP)
GetY	Возращает координату Y текущей позиции (CP)
GraphErrorMsg	Возвращает строку сообщения об ошибке для заданного кода ErrorCode
GraphResult	Возвращает код ошибки для последней графической операции
ImageSize	Возвращает число байтов, требуемое для заполнения прямоугольной области экрана InstallUserDriver.Устанавливает пользовательский драйвер устройства в BGI-таблицу драйверов устройств
InstallUserFont	Устанавливает новый шрифт который не встроен и BGI-систему
RegisterBGIdriver	Регистрирует драйвер BC1 для графической системы
RegisterBGIfont	Регистрирует шрифт BGI для графической системы
TextHeight	Возвращает высоту строки в пикселях
TextWidth	Возвращает ширину строки в пикселях

Таблица Д.3 - Работа с текстовыми блоками.

Основные команды	Функции
Ctrl+K,B	В отметить начало блока в позиции курсора
Ctrl+K,K	отметить конец блока в позиции курсора (новой).
Ctrl+K,T	отметить в качестве блока слово слева от курсора.
Ctrl+K,L	отметить в качестве блока строку, в которой курсор.
Shift+End	- отметить блок от курсора до конца строки.
Shift+ Home	отметить блок от курсора до начала строки.
CtrI+K,P	отпечатать текстовый блок (включите принтер).
Ctrl+K,V	переместить блок в новую позицию, заданную курсором.
Ctrl+K,C	копировать блок в позицию, заданную курсорам.
Ctrl+K,Y	удалить текстовый блок (или Ctrl+Del)
Ctrl+K,I	сместить блок вправо на одну позицию
Ctrl+K,U	сместить блок влево на одну позицию
CtrI+K,H	снять световое выделение блока/вновь высветить блок
CtrI+K,W	записать блок на диск (будет запрос, как назвать файл).
Ctrl+K,R	вставить текстовый файл, хранимый на диске, в позицию курсора
Ctrl+Q,L	восстановить испорченную или стертую строку (если курсор не переводился на другую строку).
Из одного окна редактора в другое можно переносить:	
Ctrl+Ins	копировать блок в так называемый "карман".
F6	перейти в другое окно (или New - открыть другое окно).
Shift+Ins	копировать блок из "кармана" в окно
Shift+Del	поместить блок в "карман", удалив его из окна
Команды, используемые при отладке программы:	
F9	запускает компиляцию программы в активном окне.
Ctrl+F9	запускает программу на автоматическое выполнение
F4	запускает программу на выполнение до строки с курсором.
F7	выполняет строку программы с заходом в блоки
F8	выполняет строку программы без захода в блоки
Ctrl+F2	начальная установка программы после прерванного выполнения
Ctrl+F7	добавляет выражение в окно просмотра (окно будет создано).
Ctrl+F8	устанавливает или уничтожает точку прерывания в строке с курсором
Ctrl+F5	открывает экран пользователя (там данные и результаты).
F6-	переходит к другому окну, становящемуся активным
Alt+F3	закрывает активное окно
Del	уничтожает строки в окне просмотра, когда оно активно.