1 курс

ПЛАН – КОНСПЕКТ
проведения лекционного занятия, практического занятия № 26
по дисциплине «Информатика»

Раздел 4. «Основы алгоритмизации и программирования.»

Тема 4.3.: «Графический режим.»

часть 1

Подготовил: преподаватель
В.Н. Борисов

Рязань 2023

Вопросы занятий:

1. Работа в графическом режиме.
2. Графические процедуры и функции.
3. Параметры графических объектов и способы их изменения.
4. Составление программы, использующей графические процедуры и функции (практическое занятие № 26, теоретическая часть, выполнение практического задания).

Время проведения лекционного, практического занятия – 6 часов.

Первый вопрос: Работа в графическом режиме.

Второй вопрос: Графические процедуры и функции.

Третий вопрос: Параметры графических объектов и способы их изменения.

Четвертый вопрос: Составление программы, использующей графические процедуры и функции (практическое занятие № 26, теоретическая часть, выполнение практического задания).

Сведения по вопросам 1-4 смотри на следующих страницах.

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

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

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

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

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

Теоретические положения

Графический режим

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

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

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

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

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

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

Таблица 1

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

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

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

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

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

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

Очистка графического экрана

Процедура **CLEARDEVICE.**

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

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

Формат процедуры без параметров.

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

Построение контуров фигур

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

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

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

Таблица 2

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

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

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

Таблица 3

Константа	Значение	Описание
NormWidth	1	Нормальной толщины

ThickWidth	3	Удвоенной толщины

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

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

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

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

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

Контуры элементарных фигур

Точка

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

Отрезок

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

Прямоугольник

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

Окружность

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

Дуга окружности

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

Угол задается в градусах (от 0 до 360).

Эллипс и дуга эллипса

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

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

Построение закрашенных изображений

В Турбо Паскале заполненные изображения строятся в два этапа:

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

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

Таблица 4

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

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

2 этап. Построить изображение.

Список элементарных закрашенных изображений с описанием параметров приводится ниже

Закрашенный прямоугольник

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

Закрашенный эллипс

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

Закрашенный сектор эллипса

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

Закрашенный сектор окружности

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

Закрашивание замкнутой области

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

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

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

Порядок выполнения работы

1. Уточните номер своего варианта.

2. Ознакомьтесь со списком графических операторов в теоретических положениях.

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

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

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

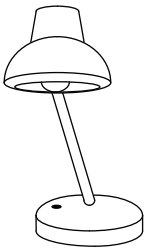
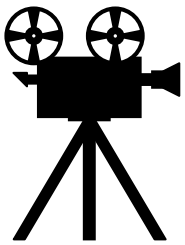
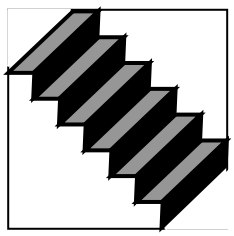
5. Отладьте программу (исключите все сообщения об ошибках).

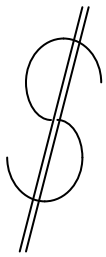
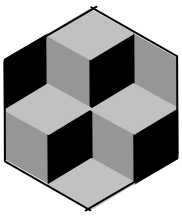
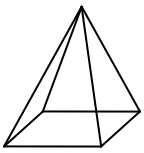
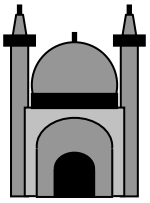
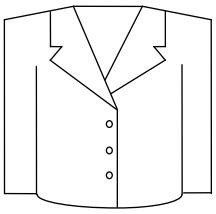
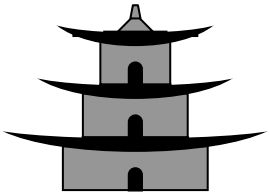
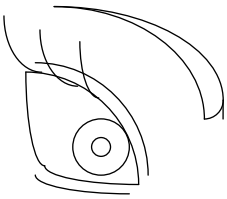
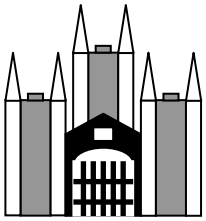
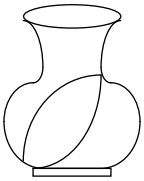
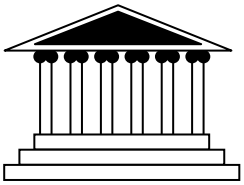
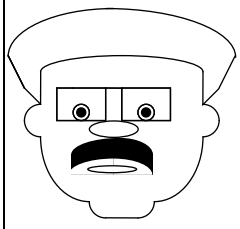
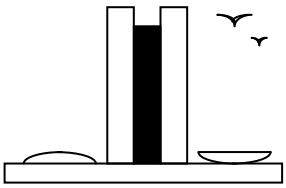
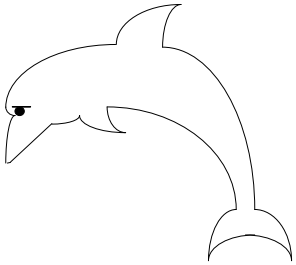
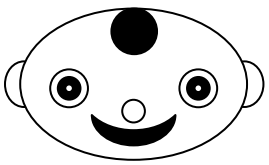
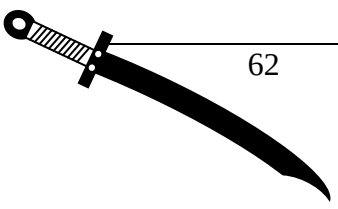
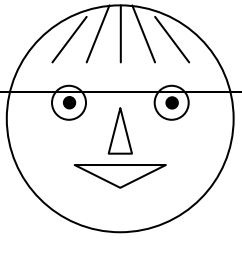
6. В текстовом редакторе WORD или рукописно оформите отчет в соответствии с содержанием.

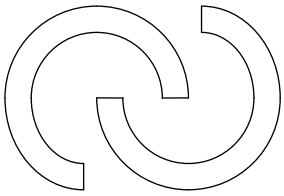
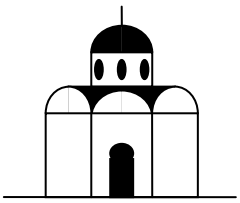
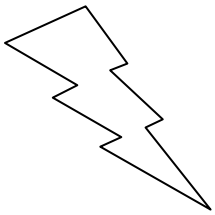
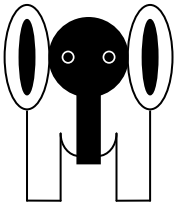
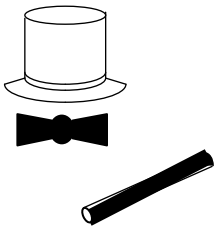
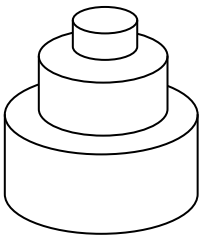
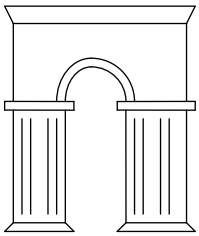
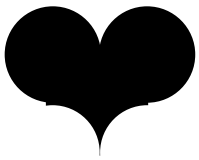
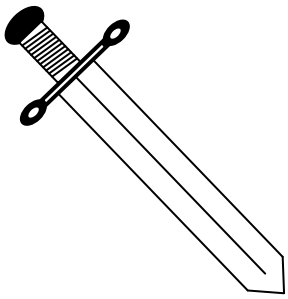
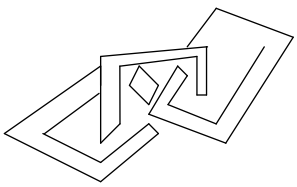
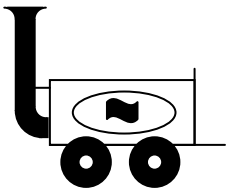
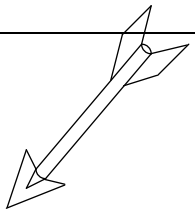
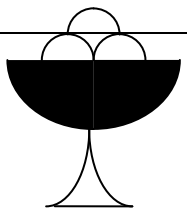
Содержание отчета

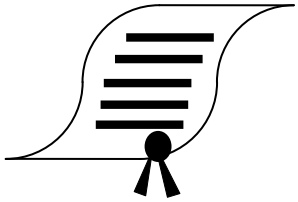
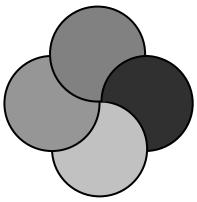
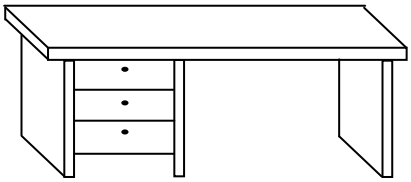
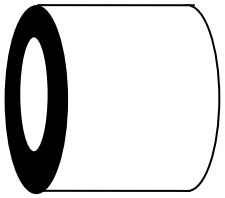
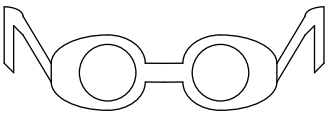
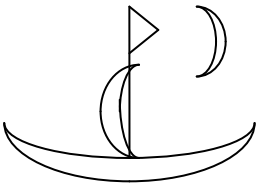
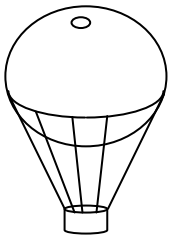
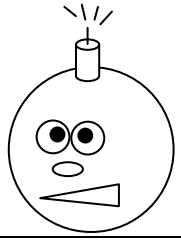
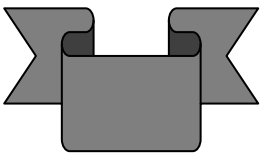
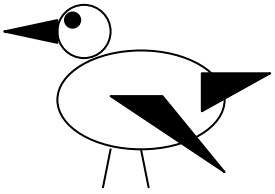
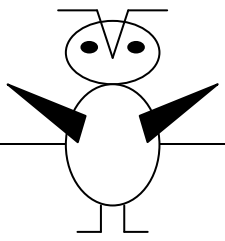
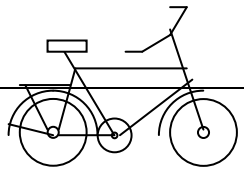
1. Титульный лист.
2. Задания варианта.
3. Описание использованных в работе операторов.
4. Программные единицы.

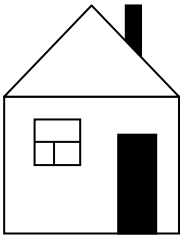
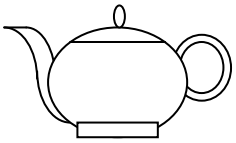
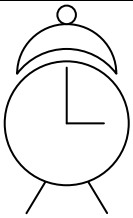
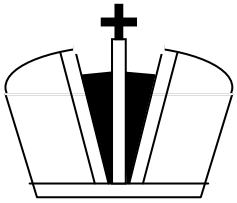
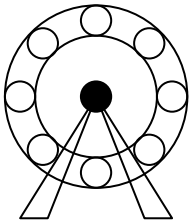
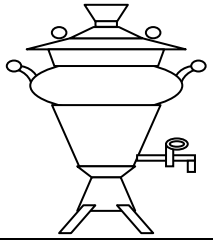
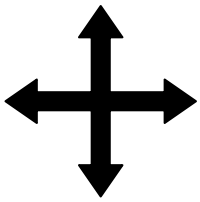
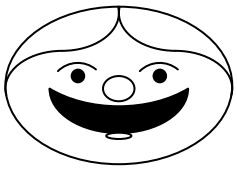
Варианты задач

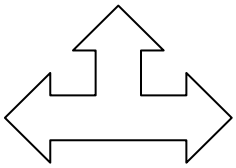
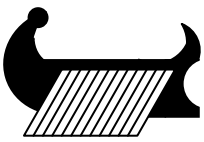
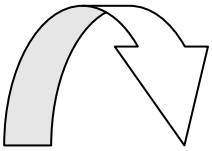
Вариант 1		
Вариант 2		

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

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

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

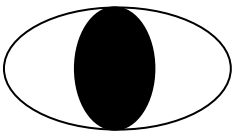
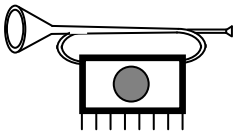
Вариант 25		
Вариант 26		
Вариант 27		
Вариант 28		

Вариант 29		
Вариант 30		

Решение типового варианта

Рассмотрим пример оформления работы.

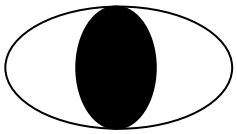
Пусть вариант студента 31 содержит следующие задания.

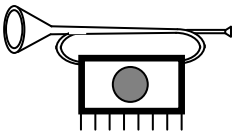
Вариант 31.		

Сформируем отчет к лабораторной работе 3.

1. Титульный лист.

2. Сформулируем задание варианта 31.

1. 

2. 

3. Описание использованных в работе операторов.

В программе будут использованы операторы.

Отрезок

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

Закрашенный эллипс

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

Прямоугольник

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

4. Программные единицы.

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

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

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

Список литературы

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

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

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

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

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

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

Вопросы для самопроверки

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

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

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

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

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

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

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

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

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

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

Далее по вопросу 4:

Работа в графическом режиме в ABC Pascal.net.

Модуль GraphABC содержит константы и функции для работы с цветами. Тип ColorType, описывающий цвет, определен следующим образом: type ColorType=integer;

Стандартные цвета задаются символическими константами:

clBlack – черный
clPurple – фиолетовый
clWhite – белый
clMaroon – темно-красный
clRed – красный
clNavy – темно-синий
clGreen – зеленый
clBrown – коричневый
clBlue – синий
clSkyBlue – голубой
clYellow – желтый
clCream – кремовый
clAqua – бирюзовый
clOlive – оливковый
clFuchsia – сиреневый
clTeal – сине-зеленый
clGray – темно-серый
clLime – ярко-зеленый
clMoneyGreen – цвет зеленых денег
clLtGray – светло-серый
clDkGray – темно-серый
clMedGray – серый
clSilver – серебристый

Для работы с цветами используются следующие функции:

function RGB(r,g,b: integer): ColorType; - возвращает целое значение, являющееся кодом цвета, который содержит красную, зеленую и синюю составляющие с

интенсивностями R,G и B соответственно (R,G и B – целые в диапазоне от 0 до 255, причем, 0 соответствует минимальной интенсивности, 255 – максимальной).

function GetRed(color: ColorType): integer; - выделяет красный цвет интенсивностью (целое число от 0 до 255); function GetGreen(color: ColorType): integer; - выделяет зеленый цвет интенсивностью (целое число от 0 до 255);

function GetBlue(color: ColorType): integer; - выделяет синий цвет интенсивностью (целое число от 0 до 255).

Составление программы, использующей графические процедуры и функции в ABC Pascal.net (программы, рисующие фигурку и домик).

```
Program Figurka;
uses GraphABC;
var w,r,c: integer;
begin
SetWindowSize(500,500); //задаём размер графического окна
SetPenWidth(3); //устанавливаем стиль пера
SetBrushColor(clFuchsia); //устанавливаем цвет кисти
Circle(225,160,50); //рисуем окружность
Line(225,160,225,180); //рисуем линии
Line(210,190,240,190);
Line(225,210,225,250);
Line(100,100,200,260);
Line(200,260,400,260);
Line(210,350,200,480);
Line(240,350,250,480);
Rectangle(200,230,250,350); //рисуем прямоугольник
SetBrushColor(clLime);
FillRect(0,480,500,500); //рисуем закрашенный прямоугольник
SetBrushColor(clWhite);
Circle(205,150,10);
Circle(245,150,10);
end.
```

```
Program Domik;
 uses graphABC; //подключение модуля graphABC
 begin
 SetWindowWidth(800); //ширина окна программы
```

```
SetWindowHeight(600); //высота окна программы
SetFontStyle(fsBold); //жирный стиль шрифта
SetFontSize(18); //размер шрифта
SetFontColor(clRed); //цвет шрифта
TextOut(100,100,'Домик'); //текст
Rectangle(200,300,600,600); //дом
Circle(400,225,40); //круг
SetBrushColor(clAqua); //цвет заливки окна
FillRect(300,400,500,500); //процедура заливки окна
Rectangle(300,400,500,500); //окно
Line(400,400,400,500); //окно
Line(300,450,500,450); //окно
Line(200,300,400,150); //крыша
Line(400,150,600,300); //крыша
Line(480,210,480,160); //труба
Line(480,160,520,160); //труба
Line(520,160,520,240); //труба
end.
```