КИПР

ТУСУР

Кафедра конструирования
и производства радиоаппаратуры

УТВЕРЖДАЮ
Заведующий кафедрой КИПР
_____**В.Н. ТАТАРИНОВ**
“___” _____2012 г.

Работа с экраном в графическом режиме

Лабораторная работа по дисциплинам **«**Информатика**»** для студентов специальностей 211000.62 «Конструирование и технология электронных средств» (бакалавриат) и 162107.65 «Информатика и информационные технологии» (специалитет)

Разработчик:
Доцент кафедры КИПР
_____**Ю.П. Кобрин**

Томск 2012

Оглавление

1 Цели работы

Целью лабораторной работы является овладение основными приемами работы в графическом режиме:

- ✓ инициализации графического режима в ***Borland Pascal***;
- ✓ рисования графика функции с помощью точек, отрезков линий, прямоугольников и т.п.;
- ✓ вывода текста на экран в графическом режиме;
- ✓ вывода на печать графика, представленного на экране.

2 Порядок выполнения работы

В ходе выполнения этой работы следует:

1. Изучить описание лабораторной работы, обратив особое внимание на особенности работы в графическом режиме в ***Borland Pascal***. При необходимости использовать дополнительную литературу [1 - 9].
2. Ответить письменно на контрольные вопросы.
3. Войти в свой личный каталог, и настроить интегрированную среду ***Borland Pascal*** для последующей работы. Записать файл конфигурации в личный каталог.
4. Внимательно проштудировать учебный пример и по возможности максимально использовать его в своей задаче.
5. Ввести и отладить разработанную программу.
6. Продемонстрировать работоспособность программы для различных вариантов исходных данных.
7. Оформить отчет по лабораторной работе и защитить его у преподавателя.

3 Контрольные вопросы

Ответьте письменно на следующие контрольные вопросы:

1) Как отображается информация в текстовом и графическом режимах?

2) Назовите основные возможности стандартного модуля ***Graph***.

3) Как перейти в графический режим и обратно в текстовый? Поясните назначение и параметры подпрограмм инициализации и отмены графического режима *(**InitGraph*** и ***CloseGraph**)*.

4) Какая функция позволяет выполнять обработку ошибок графического режима?

5) Назовите основные подпрограммы и правила их использования для вывода текста в графическом режиме.

6) Поясните назначение и параметры подпрограмм установления стиля и цвета линий и текстов , а также вывода текстов *(**SetLineStyle***, ***SetTextStyle***, ***SetColor*** и ***OutTextXY**)*.

7) Назовите основные подпрограммы и правила их использования для вывода линий и точек *(**Line***, ***LineTo***, ***LineRel***, ***PutPixel**)*.

4 Отчетность

Отчет должен быть выполнен в соответствии с [10] и состоять из следующих разделов:

1) Тема и цель работы.

2) Индивидуальное задание.

3) Схема алгоритма решения задачи.

4) Текст программы и вводимые тестовые исходные данные.
5) Откомпилированный текст программы-заготовки (в электронном виде).
6) Результаты выполнения программы.
7) Ответы на контрольные вопросы.
8) Выводы.

При защите отчета по работе для получения зачета студент должен:

- ✓ *уметь* отвечать *на контрольные вопросы*;
- ✓ обосновать структуру выбранного алгоритма;
- ✓ уметь пояснять работу программы и доказать ее работоспособность;
- ✓ продемонстрировать *навыки работы в среде* ***Borland Pascal***.

5 Графический режим работы с экраном

5.1 Основные понятия

Любое изображение на экране дисплея формируется в графическом режиме растровым способом в одной или нескольких страницах видеопамяти из множества отдельных светящихся точек – ***пикселей*** (*PIxEL - Picture ELement* - элемент растра), которые можно закрасить в тот или иной цвет.

Графический режим в компьютерах (количество пикселей в строке и столбце, цветов, уровней цветности и яркости и т.п.) зависит от типа электронной платы - *адаптера графического режима*[1] (*CGA, EGA, VGA, SuperVGA* и т.д.).

Программно работу графического адаптера поддерживает специальная программа – ***драйвер***[2]. Драйвер определяет - как при формировании графического изображения будут интерпретироваться данные, размещенные в видеопамяти.

В качестве курсора в графическом режиме используется *невидимый текущий указатель* (аналогичен текстовому курсору). **Ошибка! Источник ссылки не найден.** показывает, что координаты курсора и каждого пикселя определяются относительно *левого верхнего угла* экрана, который имеет координаты ***0,0***. Координата ***X*** увеличивается по горизонтали *слева направо*, а координата ***Y*** – по верти-

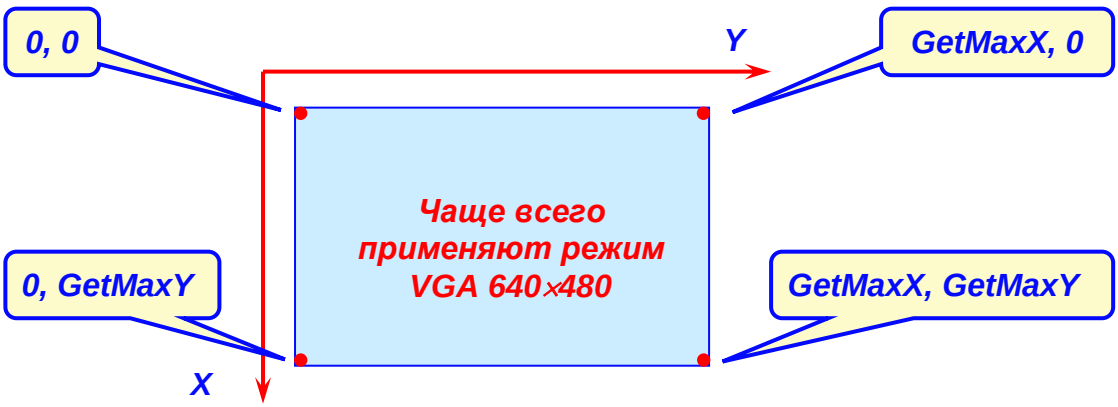

Рисунок 5.1 - Координаты точек экрана в графическом режиме

[1] Например, наиболее популярный в настоящее время графический адаптер ***VGA*** (*Video Gate Array* – видео графическая матрица) обеспечивает размещение на экране 640 × 480 пикселей.

[2] Для программ на Паскале применяют графические драйверы, разработанные фирмой ***Borland*** практически для всех типов адаптеров. Обычно они размещены в отдельном подкаталоге ***BGI*** в виде файлов с расширением ****.BGI*** (от англ. ***Borland Graphics Interface*** - графический интерфейс фирмы ***Borland***).

кали *сверху вниз*.

GetMaxX и ***GetMaxY*** - стандартные функции модуля ***Graph***, которые возвращают соответственно максимальные координаты по осям ***X*** и ***Y*** в зависимости от текущего режима видеоадаптера. Другие необходимые подпрограммы для графического режима также находятся в модуле ***Graph***. С помощью инструкции ***Uses Graph*** его следует подключить к разрабатываемой графической программе. Модуль ***Graph*** содержит процедуры и функции, которые предоставляют пользователю разнообразные возможности управления графическим экраном, в том числе рисование дуг, окружностей, прямых и ломаных линий, прямоугольников, многоугольников, закраску фигур требуемым цветом, их штриховку и вывод текстов.

5.2 Инициализация графического режима

Чтобы программа могла выводить на экран графику, нужно инициализировать графический режим работы. Инициализацию графического режима выполняет процедура ***InitGraph***:

InitGraph (GraphDriver, GraphMode, PathDriver);

Переменная ***GraphDriver*** (тип *integer*) определяет используемый программой драйвер видеоадаптера, переменная ***GraphMode*** (тип *integer*) - режим работы видеосистемы (*табл. 5.1*), а переменная (или текстовая константа) ***PathDriver*** (тип *string*) – синтаксически корректный маршрут к каталогу, в котором находятся файлы графических драйверов (****.BGI***).

Таблица 5.1- Режимы работы адаптера VGA

Режим (значение *GraphMode*)	Константа режима	Разрешение	Число цветов	Число видеостраниц
0	***VGALo***	640 × 200	16	4
1	***VGAMed***	640 × 350	16	2
2	***VGAHi***	640 × 480	16	1

Чтобы использовать широко распространенные в настоящее время адаптеры ***VGA*** (или ***SuperVGA***) (их работа поддерживается драйвером ***EgaVga.bgi***) следует в процедуре ***InitGraph*** указать номер графического режима ***GraphDriver = 9***[3]***.***

Если программа должна быть рассчитана на работу с любым адаптером, то для его автоматического определения перед инициализацией графического режима следует присвоить ***GraphDriver := Detect.*** Присваивать какое-либо значение переменной режима в этом случае не требуется

Внутренние ошибки модуля ***Graph*** возвращаются функцией ***GraphResult***, показывающей состояние последней графической операции (см. *табл. 5.2*). Например, если инициализация графического режима ***InitGraph*** выполнена успешно, то функция ***GraphResult*** возвращает значение ***grOk = 0***, в противном случае возвращается номер ошибки графического режима[4]. Отметим, что функция ***GraphResult***

---

[3] Если в компьютере установлен адаптер ***SVGA***, то при инициализации графического режима в качестве значения параметра ***GraphDriver*** следует использовать константу ***Vga.***

[4] Чаще всего неверно указывается путь к драйверу графического режима. Чтобы компилятор Паскаля нашел эти файлы, надо в режиме *Options/Directories/Unit directories* указать каталог, в котором размещены файлы драйверов. Например: ***D:\BP7\UNITS;D:\BP7\BGI***

после обращения к ней сбрасывает код ошибки в ***0***. Следовательно, необходимо сохранить значение кода ошибки с помощью вспомогательной переменной, чтобы затем использовать его для проверки.

Таблица 5.2 - Коды ошибок, возвращаемые функцией ***GraphResult***

Константа ошибки графики	Код ошибки	Соответствующее сообщение об ошибке
grOk	***0***	Нет ошибки
grNoInitGraph	***-1***	Графика не инициализирована
grNotDetected	***-2***	Графические средства не найдены
grFileNotFound	***-3***	Файл не найден
grInvalidDriver	***-4***	Недопустимый драйвер
grNoLoadMem	***-5***	Драйвер не загружен
grNoScanMem	***-6***	Ошибка при просмотре памяти
grNoFloodMem	***-7***	Ошибка при закраске
grFontNotFound	***-8***	Шрифт не найден
grNoFontMem	***-9***	Шрифт не загружен в память
grInvalidMode	***-10***	Недопустимый режим
grError	***-11***	Ошибка графики
grIOError	***-12***	Ошибка ввода-вывода графики
grInvalidFont	***-13***	Недопустимый файл шрифта
grInvalidFontNum	***-14***	Недопустимый номер шрифта

Сообщение об ошибке на английском языке можно вывести с помощью оператора

WriteIn(GraphErrorMsg(Error));

где ***Error*** – вспомогательная переменная типа *integer* с возвращенным номером ошибки.

Для восстановления алфавитно-цифрового режима работы видеосистемы компьютера перед завершением работы программы необходимо *обязательно* вызвать процедуру ***CloseGraph***.

Пример12.1. Перевести графическую систему компьютера в режим с разрешением 640 × 480 точек и нарисовать в центре экрана 5 концентрических окружностей.

```
{Демонстрация инициализации и завершения графического режима}
program Primer12_1;
uses
  Graph; {Подключаем модуль с подпрограммами работы с графикой}
var
  GraphDriver: integer;   {номер типа адаптера и драйвера графического режима}
  GraphMode: integer;     {номер графического режима адаптера}
  PathDriver: string;     {путь к каталогу драйвера графического режима}
  Error: integer;         {результат инициализации графического режима}
  Radius: integer;        {радиус окружности}
begin
  GraphDriver  : = Detect;          {режим автоопределения типа драйвера}
  GraphMode := VgaHi;               {разрешение 640×480}
  PathDriver := 'd:\BP7\BGI';       {если драйвер находится в каталоге d:\BP7\BGI}
  InitGraph(GraphDriver, GraphMode, PathDriver); {переход в графический режим}
```

```
  Error  := GraphResult;     {Определяем результат инициализации графического
                              режима}
  if Error  <> grOk          {grOk - константа безаварийного перехода в графический
                              режим}
     then begin {Ошибка инициализации графики}
            Writeln ('Ошибка инициализации графического режима:');
            Writeln(GraphErrorMsg(Error)); {Печать сообщения о характере ошибки}
            Writeln ('Для завершения работы нажмите <Enter>');
            Readln;
            Halt              {Аварийное завершение работы программы}
          end;

  SetBkColor(Blue);         {Установим голубой цвет фона экрана}
  Writeln('Инициализация графического режима выполнена.');

  SetColor(Yellow);    {Установим желтый цвет вывода символов и окружностей}
  OutTextXY(200, 150,'Пять окружностей в центре экрана');
  for Radius := 1 to 5 do  {Цикл вывода окружностей}
    Circle(GetMaxX div 2, GetMaxY div 2, Radius * 10); {Функции GetMaxX и
                              GetMaxY возвращают максимальное число точек по осям X и Y}
  Writeln ('Для завершения работы нажмите <Enter>');
  Readln;
  CloseGraph {восстановление алфавитно-цифрового режима работы}
end.
```

5.3 Некоторые особенности работы с модулем *Graph*

Таблица 5.3 – 5.8 знакомят с примерами использования некоторых наиболее важных подпрограмм модуля ***Graph*** (неполный перечень возможностей).

Таблица 5.3 - Подпрограммы управления видеорежимами модуля ***Graph***

Заголовок подпрограммы	Назначение	Пример	Примечание
DetectGraph(var GraphDriver, GraphMode: integer)	Распознает аппаратуру и определяет, какие номера графического драйвера (***GraphDriver***) и режима (***GraphMode***) необходимо использовать.	Режим автоопределения: ***DetectGraph(GraphDriver,*** GraphMode***);***	Может применяться до инициализации графического режима
GetMaxX	Функция возвращает для текущего графического драйвера и режима самую правую колонку (разрешение по оси ***X***).	Максимальная координата ***X:*** ***XMax := GetMaxX;***	

Заголовок подпрограммы	Назначение	Пример	Примечание
GetMaxY	Функция возвращает для текущего графического драйвера и режима самую нижнюю строку (разрешение по ***Y***).	Максимальная координата ***Y:*** ***YMax := GetMaxY;***	
InitGraph(var GraphDriver, GraphMode: integer; DriverPath: string)	Инициализирует графическую систему и переводит аппаратуру в графический режим.	Из каталога *U:\BP7\BGI* загружается драйвер автоматически определяемого типа, и устанавливается разрешение 640 × 480 точек: ***GraphDriver := Detect;*** ***GraphMode := VGAHi;*** ***InitGraph(GraphDriver, GraphMode,'U:\BP7\BGI');***	***GraphDriver -*** номер типа загружаемого драйвера; ***GraphMode -*** номер режима графики; ***DriverPath -***путь к драйверу. Если задана пустая строка '' – считается, что драйвер находится в текущем каталоге.
CloseGraph	Завершает все действия с графикой. Графический драйвер удаляется из динамической памяти.	Завершение графического режима: ***CloseGraph;***	Возвратиться в графический режим можно только путем повторной его инициализации
SetGraphMode(GraphMode: integer)	Переключает систему в графический режим, указанный ***GraphMode*** и очищает экран.	Устанавливается разрешение 640 × 200 точек: ***GraphMode := VGALo;*** ***SetGraphMode(GraphMode);***	Используется только после инициализации графического режима, т.е. графический драйвер уже должен находиться в памяти
RestoreCrtMode	Возвращает систему в текстовый режим, установленный до инициализации графики	***RestoreCrtMode;***	Графический драйвер остается в памяти
GetGraphMode	Функция возвращает текущий графиче-	Переключение в графический режим с восстановлением прежних уста-	Обычно используется после перехода в тек-

Заголовок подпрограммы	Назначение	Пример	Примечание
	ский режим.	новок: ***SetGraphMode(GetGraphMode);***	стовый режим с помощью процедуры ***RestoreCrtMode***
SetViewPort(x1, y1, x2, y2: Integer; Clip: Boolean)	Задание параметров текущего окна для графического вывода: ***0 <= x1 < x2, 0 <= y1 < y2.***	Задает размеры графического окна, перемещает указатель текущей позиции в координаты (**0, 0**) окна, части фигур могут выходить за пределы окна: ***SetViewPort(10, 10, 20, 20, false);***	***x1, y1, x2, y2*** – координаты верхнего левого и правого нижнего углов окна. ***Clip*** – ограничитель фигур. Если ***Clip = true***, то все построения проводятся лишь в пределах окна
ClearViewPort	Очищается текущее графическое окно.	***ClearViewPort;***	Указатель текущей позиции перемещается в координаты (**0, 0**),

Таблица 5.4 - Подпрограммы обработки ошибок графического режима модуля ***Graph***

Заголовок подпрограммы	Назначение	Пример	Примечание
GraphResult	Функция возвращает код ошибки для последней графической операции.	Сохранение результата инициализации графического режима. Если ***ErrorCode = 0*** – ошибок нет. ***InitGraph(GraphDriver, GraphMode, ‘’);*** ***ErrorCode := GraphResult;***	Используется после процедур, вызов которых может привести к появлению ошибок (неправильная установка режимов, координат и т.п.)
GraphErrorMsg(ErrorCode)	Для заданного кода ошибки функция возвращает строку сообщения о причине ошибки (на английском	***If ErrorCode <> 0 then Writeln(‘Ошибка графики’, GraphErrorMsg(ErrorCode));***	Значение переменной ***ErrorCode*** получено с помощью функции ***GraphResult***

языке).

Таблица 5.5 - Подпрограммы построения простейших изображений модуля ***Graph***

Заголовок подпрограммы	Назначение	Пример	Примечание
PutPixel(X, Y: Integer; Color: Word)	Вывод точки (пикселя) в координатах ***X***, ***Y*** и с цветом ***Color***.	Выводится красная точка, координаты (10, 15) ***PutPixel(10, 15,Red);***	
Line(x1, y1, x2, y2: Integer)	Рисует прямую линию с текущим типом и цветом из точки (***x1, y1***) в (***x2, y2***).	Рисует линию между точками с координатами **(100, 100)** и **(200, 200)** ***Line(100, 100, 200, 200);***	
LineRel(dX, dY: Integer)	Рисует прямую линию текущего типа и цвета до точки, представляющей собой относительное расстояние от текущего указателя.	Рисует линию между точками с координатами **(100, 100)** и **(200, 200)** ***MoveTo(100, 100);*** ***LineRel(100, 100);***	***dX, dY*** – приращения соответствующих координат
LineTo(X, Y: Integer)	Рисует линию из текущего положения в точку (***X, Y***).	Рисует линию между точками с координатами **(100, 100)** и **(200, 200)** ***MoveTo(100, 100);*** ***LineTo(200, 200);***	***X, Y*** – координаты конца отрезка линии
Circle(X, Y: Integer; Radius: Word)	Рисует окружность радиусом ***Radius*** текущим цветом с центром в точке (***X, Y***).	Рисует окружность с координатами **(100, 100)** и радиусом 50 ***Circle(100, 100, 50);***	
Rectangle((x1, y1, x2, y2)	Рисует прямоугольник, используя текущий тип линии и цвет.	Рисует прямоугольник с координатами (100, 100) и (200, 200) ***Rectangle(100, 100, 200, 200);***	***x1, y1, x2, y2*** – координаты верхнего левого и правого нижнего углов прямоугольника.

Заголовок подпрограммы	Назначение	Пример	Примечание
Arc(X, Y: Integer; StAngle, EndAngle, Radius: Word)	Рисует дугу окружности текущего цвета.	Рисует дугу окружности с координатами центра (100, 100) и радиусом 50 в секторе от 0^o до 90^o от горизонтальной оси против часовой стрелки ***Arc(100, 100, 0, 90, 50);***	(***X, Y***) берется в качестве центра окружности; ***StAngle, EndAngle, Radius*** – начальный, конечный углы и радиус соответственно.

Таблица 5.6 - Подпрограммы установки параметров линий модуля ***Graph***

Заголовок подпрограммы	Назначение	Пример	Примечание
SetLineStyle(LineStyle: Word; Pattern: Word; ThickNess: Word)	Устанавливает текущий тип линии ***LineStyle***[5] и ее ширину ***ThickNess***[6].	Устанавливается режим толстой пунктирной линии ***SetLineStyle(DottedLn , 0, ThickWidth);***	***Pattern*** – переменная, задающая тип линии при ***LineStyle = UserBitLn;*** представляется в виде двухбайтового числа, каждый бит которого равен **1**, если нужно нарисовать очередной пиксель, и **0** – в противном случае.
SetPalette(ColorNum: Word; Color: ShortInt)	Изменяет один цвет палитры[7].	Меняет **0** компонент палитры (по умолчанию черный) на светло-голубой} ***SetPalette(0, LigthCyan);***	***ColorNum*** - номер компоненты палитры; ***Color*** – задаваемый цвет.
SetBkColor(ColorNum: Word)	Устанавливает текущий фоновый цвет.	Меняет цвет фона (по умолчанию черный) на	***ColorNum*** – номер цвета в палитре

[5] Параметры ***LineStyle: SolidLn*** – сплошная, ***DottedLn*** – пунктирная, ***CenterLn*** – штрихпунктирная, ***DashedLn*** – штриховая, ***UserBitLn*** – заданная пользователем.

[6] Параметры ***ThickNess: NormWidth*** – нормальная, ***ThickWidth*** – толстая.

[7] ***Палитра*** – совокупность одновременно доступных цветов..

Заголовок подпрограммы	Назначение	Пример	Примечание
		синий ***SetBkColor(Blue);***	
SetCol-or(ColorNum: Word)	Устанавливает текущий цвет рисунка.	Меняет цвет линии (по умолчанию белый) на желтый ***SetColor(Yellow);***	***ColorNum*** – номер цвета в палитре
GetBkColor	Функция возвращает текущий фоновый цвет.	В ***BkColor*** запоминается текущий фоновый цвет ***BkColor := GetBkColor;***	
GetColor	Функция возвращает текущий цвет рисунка.	В ***Color*** запоминается текущий цвет линии ***Color := GetColor;***	

Таблица 5.7 - Подпрограммы управления текущим указателем модуля ***Graph***

Заголовок подпрограммы	Назначение	Пример	Примечание
MoveTo(X, Y: Integer)	Перемещает текущий указатель в точку (***X, Y***).	Перемещает текущий указатель в точку (***100, 100***) ***MoveTo(100. 100);***	
MoveRel(dX, dY: Integer)	Перемещает текущий указатель на расстояние, являющееся относительным расстоянием от текущей позиции.	Перемещает текущий указатель в точку (***120, 80***) ***MoveTo(100. 100);*** ***MoveRel(20, -20)***	***dX, dY*** – приращения соответствующих координат

Таблица 5.8 - Подпрограммы вывода текста модуля ***Graph***

Заголовок подпрограммы	Назначение	Пример	Примечание
OutText(TextString: String)	Посылает строку ***TextString*** на устройство вывода, начиная с текущего указателя.	Вывод текста «Кипр» с позиции (***100, 100***) ***MoveTo(100. 100);*** ***OutText('Кипр');***	
OutTextXY(X, Y: Integer; TextString: String)	Посылает строку ***TextString*** на устройство выво-	Вывод текста «Кипр» с позиции (***100, 100***) ***OutText(100, 100,***	

Заголовок подпрограммы	Назначение	Пример	Примечание
	да, начиная с координат (*X, Y*) .	*'Кипр');*	
SetTextJustify(Horiz, Vert: Word)	С помощью *OutText* и *OutTextXY* устанавливает значения для выравнивания текста. *Horiz, Vert* – переменные, задающие выравнивание по горизонтали и вертикали.	Текст выравнивается по горизонтали - к левой границе, по вертикали – по центру *SetTextJustify(LeftText, CenterText);*	
SetTextStyle(Font, Direction: Word; CharSize: Word)	Задает текущий текстовый шрифт *Font*, его расположение *Direction* и коэффициент размера символа *CharSize*.	Тип шрифта – по умолчанию, текст в горизонтальной строке, размер символов увеличен в 2 раза *SetTextStyle(DefaultFont, HorizDir, 2);*	

Для получения координат указателя графического вывода по осям X и Y можно обратиться к функциям ***GetX*** и ***GetY***. Для перемещения указателя в нужную точку экрана следует вызвать процедуру ***MoveTo***. Например, операторы

```
MoveTo(305, 240); {Перемещаем  текущий указатель в координаты (305, 240)}
OutText('Borland Pascal 7.0');
```

выводят сообщение примерно в центре экрана.

Любой чертеж, схема, картинка могут рассматриваться как ***совокупность графических примитивов***: точек, линий, окружностей, дуг и т.п. Следовательно, чтобы на экране появилась нужное изображение, программа должна обеспечить вычерчивание (вывод) графических примитивов, составляющих эту картинку. Вычерчивание на экране графических примитивов осуществляют соответствующие процедуры модуля ***Graph***: ***PutPixel,. Line, LineRel, LineTo, Circle, Rectangle*** и т.п.

Для смены текущего цвета вывода, перед вычерчиванием графических примитивов применяют процедуру ***SetColor***, в которую в качестве параметра передается номер требуемого цвета палитры (см. *табл. 5.9*).

Таблица 5.9 - Номера цветов в стандартной палитре модуля ***Graph***

Цвет	Имя константы	Номер цвета
Черный	***Black***	***0***
Синий	***Blue***	***1***
Зеленый	***Green***	***2***
Голубой	***Cyan***	***3***
Красный	***Red***	***4***
Фиолетовый	***Magenta***	***5***
Коричневый	***Brown***	***6***

Цвет	Имя константы	Номер цвета
Светло-Серый	LightGray	7
Темно-Серый	DarkGray	8
Светло-Синий	LightBlue	9
Светло-Зеленый	LightGreen	10
Светло-Голубой	LightCyan	11
Розовый	LightRed	12
Светло-Фиолетовый	LightMagenta	13
Желтый	Yellow	14
Белый	White	15

Для вывода текста в графическом режиме используется шрифт с растром 8×8 и несколько векторных шрифтов. Растровый символ задается с помощью матрицы элементов изображения. Векторный шрифт задается рядом векторов, которые указывают графической системе, как рисовать шрифт.

Шрифты хранятся каждый в отдельном файле на диске в подкаталоге BGI и должны присутствовать там во время работы (при вызове процедуры ***SetTextStyle***). Файлы шрифтов имеют расширение **.CHR* и могут загружаться с диска автоматически модулем ***Graph***. Их также можно компоновать с программой пользователя или загружать и "регистрировать" с помощью модуля ***Graph***. Размер векторного шрифта можно настроить с помощью процедуры ***SetUserCharSize***.

Полученное на экране изображение можно считать с экрана и запомнить, а затем снова воспроизвести на экран. Его можно отпечатать после запуска программы, обеспечивающей эту возможность.

6 Вывод графиков функций

6.1 Условия задания

Одной из самых известных проблем является отображение результатов проектирования в виде графиков на экране и вывод их твердой копии на принтер. Рассмотрим один из возможных вариантов решения этой проблемы на примере.

Пример 12.2. Разработать алгоритм и программу вывода на экран графика функции заданной массивами абсцисс X и ординат Y разными способами, с помощью разных стандартных процедур (PutPixel, LineTo, Circle, Rectangle, Bar). Программа должна легко настраиваться на различные пределы изменения кривой и максимально использовать площадь экрана, для чего следует выполнить автоматический выбор масштаба по осям X и Y. Вывести:

- ✓ заголовок графика функции;
- ✓ диапазоны изменения функции по оси X и Y;
- ✓ график функции, отображаемый в сетке, с оцифровкой, округленной до ближайшего допустимого значения;
- ✓ названия осей;
- ✓ вывести на печать экран с выведенным графиком функции.

Для вывода графика функции с экрана на печать необходимо предварительно запустить на выполнение из командной строки ***MS DOS*** одну из системных программ: ***prtscr.com*** или ***graphics.com.***

Для решения задачи разработаны головная программа ***Primer12_2*** и пользовательский модуль с применяемыми в ней подпрограммами ***Unit12_2***.

Большое количество комментариев позволит Вам легко разобраться с используемыми алгоритмами и адаптировать их для решения своих задач.

6.2 Листинг головной программы *Primer12_2*

```
program Primer12_2; {Вывод на экран графика функции одной переменной
                       заданной массивами абсцисс X и ординат Y
                       с  помощью  различных стандартных графических процедур:
                       PutPixel, LineTo, Circle, Rectangle, Bar.
                    Выполнить автоматический выбор масштаба по осям X и Y.
                    Вывести:
                     - заголовок графика функции;
                     - диапазоны изменения функции по оси X и Y;
                     - график функции, отображаемый в сетке, с оцифровкой,
                       округленной до ближайшего допустимого значения;
                     - названия осей}

uses  {Подключаем модули: }
  Graph,      {с подпрограммами работы с графикой}
  Unit12_2; {с подпрограммами для вывода на экран графика функции}
var
  X, Y: tMassiv; {Массивы абсцисс и ординат функции на графике}
  N: integer;    {количество расчетных точек функции}
  MinX, MaxX, MinY, MaxY: real; {Мин. и макс. значения абсцисс и ординат функции}
  NX, NY: integer;  {число отсчетов (сетки) по осям X и Y}
  DelX, DelY: real; {шаг отсчетов (сетки) по осям X и Y}

{Отображаемая на графике функция одной переменной}
function F(x: real): real; far; {дальний тип вызова}
begin
   F := 10 * sin(3 * x) * exp(0.2 * x)
end; {F}

begin {начало основного блока программы Primer12_2}

  N := 200; {число расчетных точек функции}

  {Формирование массивов абсцисс X и ординат Y функции F
   в диапазоне изменения аргумента -10 .. 10}
  Table(N, X, Y, F, -10, 10);

  {Определение масштаба по оси X}
  NX := 10; {число отсчетов (сетки) по оси X}
  Mashtab(N,          {число расчетных точек функции}
       X,             {массивы абсцисс функции}
       LengthXPix,    {длина окна графика по оси X (в пикселях)}
       NX,            {число отсчетов (сетки) по оси X}
       MinX, MaxX,    {мин. и макс. значения абсцисс функции}
       DelX,          {шаг сетки по оси X}
       MaschtabX);    {масштаб по оси X}

  {Определение масштаба по оси Y}
  NY := 7; {число отсчетов (сетки) по оси Y}
  Mashtab(N,          {число расчетных точек функции}
       Y,             {массив ординат функции}
```

```
      LengthYPix,    {длина окна графика по оси Y (в пикселях)}
      NY,            {число отсчетов (сетки) по оси Y}
      MinY, MaxY,    {мин. и макс. значения ординат функции}
      DelY,          {шаг отсчетов (сетки) по оси Y}
      MaschtabY);    {масштаб по оси Y}

  {Уточняем путь - файл драйвера графического режима находится в каталоге d:\BP\BGI}
  PathToDriver := 'd:\BP\BGI';

  {Перевод дисплея в графический режим с разрешением 640 x 480}
  GraphInit(VgaHi);

  {Вывод на экран сетки графика}
  Setka('График функции f(x) = 10 * sin(3 * x) * exp(0.2 * x)', {заголовок графика функции}
      'X', 'Y',          {надписи к осям X и Y}
      MinX, MaxX,        {мин. и макс. значения абсцисс функции}
      MinY, MaxY,        {мин. и макс. значения ординат функции}
      DelX, DelY,        {шаг отсчетов (сетки) по осям X и Y}
      MaschtabX, MaschtabY, {масштабы по осям X и Y}
      Blue,              {цвет фона графика}
      Yellow,            {цвет заголовка графика - желтый}
      LightGray,         {цвет границ окна - светло-серый}
      Cyan,              {цвет сетки и оцифровки графика - бирюзовый}
      LightCyan);        {цвет осей графика - светло-бирюзовый}

  {Вывод на экран графика функции, выполненного линиями}
  GraphFun(2,  {VarGr - вариант вывода графика:
                1 - точками,
                2 - линией,
                3 - кружками,
                4 - прямоугольниками,
                5 - гистограмма}
      MaschtabX,     {масштаб по оси X}
      MaschtabY,     {масштаб по оси Y}
      N,             {число расчетных точек функции}
      X, Y,          {массивы абсцисс и ординат функции}
      Yellow);       {цвет графика}

  Pause; {временная остановка до нажатия <<Enter>>}

  CloseGraph {восстановление алфавитно-цифрового режима работы дисплея}
end.
```

6.3 Листинг пользовательского модуля *Unit12_2*

```
{$F+}
Unit Unit12_2;      {Модуль работы с графической системой компьютера
                     для вывода на экран графиков функций}

interface {интерфейсная секция}

uses
```

```
  Graph; {Подключаем модуль с подпрограммами работы с графикой}

Const {раздел объявления констант}

  {Путь к каталогу драйвера графического режима. По умолчанию файл драйвера находится
в том же каталоге, что и программа}
  PathToDriver: string = '';
  LengthXPix: integer = 500; {Длина окна графика по оси X по умолчанию (в пикселях)}
  LengthYPix: integer = 400; {Длина окна графика по оси Y по умолчанию (в пикселях)}
  NMax = 500; {Максимальное количество точек расчета функции}

type  {раздел объявления нестандартных пользовательских типов данных}
   tMassiv = array[1..NMax] of real; {для хранения абсцисс или ординат функции}
   tFun = function(x: real): real;      {тип отображаемой на графике функции}
   tStr80 = string[80];                 {строка для заголовка графика функции}
   tStr10 = string[10];                 {строка для надписей к осям X и Y}

{Переход в графический режим}
procedure GraphInit(GrMode: integer {GrMode - номер графического режима адаптера});

{Формирование массивов абсцисс X и ординат Y функции Fun
 N - число расчетных точек в диапазоне изменения аргумента xMin..xMax
 Fun - формальное имя функции одной переменной, отображаемой на графике}
procedure Table(N: Integer; var X, Y: tMassiv; Fun: tFun; xMin, xMax: real);

{Поиск минимального MinR и максимального MaxR элементов в массиве R длиной N}
procedure MinMax(N: Integer; R: tMassiv; var MinR, MaxR: real);

{Определение масштабов по осям X или Y}
procedure Mashtab(N: integer;  {число расчетных точек функции}
                  R: tMassiv;       {массив абсцисс или ординат функции}
                  LengthRPix,       {длина окна графика по оси X или Y (в пикселях)}
                  NR: integer;      {число линий сетки по оси X или Y}
                var MinR, MaxR,     {мин. и макс. значения абсцисс или ординат функции}
                  DelR: real;       {шаг сетки по оси X или Y}
                var MaschtabR: integer); {масштаб по оси X или Y}

{Вывод на экран сетки графика}
procedure Setka(NameGraph: tStr80;    {заголовок графика функции}
                NameX, NameY: tStr10;  {надписи к осям X и Y}
                MinX, MaxX,            {мин. и макс. значения абсцисс функции}
                MinY, MaxY,            {мин. и макс. значения ординат функции}
                DelX, DelY: real;      {шаг сетки по осям X и Y}
                MaschtabX, MaschtabY: integer; {масштабы по осям X и Y}
                ColorBk,               {цвет фона графика}
                ColorNameGraph,        {цвет заголовка графика}
                ColorNameXY,           {цвет границ окна графика}
                ColorSetka,            {цвет сетки и оцифровки графика}
                ColorOsi: Byte);       {цвет осей графика}

{Вывод на экран графика функции}
```

```
procedure GraphFun(VarGr: byte; {VarGr - вариант вывода графика:
                                 1 - точками,
                                 2 - линией,
                                 3 - кружками,
                                 4 - прямоугольниками,
                                 5 - гистограмма}
                  MaschtabX,                {масштаб по оси X}
                  MaschtabY,                {масштаб по оси Y}
                  N: Integer;               {число расчетных точек функции}
                  X, Y: tMassiv;            {массивы абсцисс и ординат функции}
                  ColorGraph: byte);        {цвет отображаемой функции графика}

{Форматирование и вывод числа в заданную точку экрана в графическом режиме}
procedure Value(x, y: integer;              {координаты выводимого числа}
                R: real;                    {выводимое число}
                var x1R, x2R: LongInt;      {допустимые границы вывода числа по оси X}
                WidthR,                     {число позиций, занимаемое выводимым числом}
                Decimals: integer;          {число выводимых значимых цифр}
                Horizont: word);    {выравнивание: LeftText, CenterText, RightText}

procedure Pause; {временная остановка до нажатия <<Enter>>}

var   {раздел объявления глобальных переменных}
  GraphDriver: integer;         {номер типа адаптера и драйвера графического режима}
  GraphMode: integer;           {номер графического режима адаптера}
  Error: integer;               {результат работы в графическом режиме}
  MaschtabX,                    {масштаб по оси X}
  MaschtabY: integer;           {масштаб по оси Y}

Implementation {Секция реализации подпрограмм}

var
  x1R, x2R,     {допустимые границы вывода числа по оси X}
  xGr, yGr,     {координаты точки графика}
  x1, y1,       {координаты вернего левого  угла графика}
  x2, y2,       {координаты нижнего правого угла графика}
  x0, y0: LongInt; {координаты начала осей координат графика}

{Переход в графический режим}
{GraphMode - номер графического режима адаптера}
procedure GraphInit(GrMode: integer);
begin
  GraphDriver := Detect;        {режим автоопределения типа драйвера}
  GraphMode := GrMode;          {номер графического режима адаптера}

  InitGraph(GraphDriver, GraphMode, PathToDriver); {переход в графический режим}
  Error  := GraphResult; {Результат инициализации графического режима}
  if Error <> grOk {grOk - константа безаварийного перехода в графический режим}
    then begin {ошибка инициализации графики}
         Writeln('Ошибка инициализации графического режима:');
         Writeln ('Путь к каталогу драйвера графического режима: ', PathToDriver);
```

```
      Writeln(GraphErrorMsg(Error)); {Печать сообщения о характере ошибки}
      Writeln ('Для завершения работы нажмите <Enter>');
      Readln;
      Halt {Аварийное завершение работы программы}
    end
end;  {GraphInit}

      {Формирование массивов абсцисс X и ординат Y функции Fun
       N - число расчетных точек в диапазоне изменения аргумента xMin..xMax
       Fun - формальное имя функции одной переменной, отображаемой на графике}
procedure Table(N: Integer; var X, Y: tMassiv; Fun: tFun; xMin, xMax: real);
var
  DeltaX: real;        {Приращение аргумента X}
  argX: real;          {аргумент X}
  i: integer;          {индекс}
begin
  DeltaX := (xMax - xMin) / (N - 1);        {Приращение аргумента X}
  argX := XMin;                             {Начальное значение аргумента X}
  for i := 1 to N do            {Рассчитываем таблицу значений функции Fun}
    begin
      X[i] := argX;
      Y[i] := Fun(argX);        {Fun(X) - имя определенной заголовком процедуры функции}
      argX := argX + DeltaX  {Аргумент X для следующего цикла}
    end
end; {Table}

{Поиск минимального MinR и максимального MaxR элементов в массиве R длиной N}
procedure MinMax(N: Integer; R: tMassiv; var MinR, MaxR: real);
var
  i: integer;
begin
   MinR := R[1];
   MaxR := R[1];
   if N > 1 then
     for i := 2 to N do
       begin
         if R[i] < MinR
           then MinR := R[i];
         if R[i] > MaxR
           then MaxR := R[i]
     end
end; {MinMax}

{Определение масштабов по осям X или Y}
procedure Mashtab(N: integer;       {число расчетных точек функции}
                  R: tMassiv;       {массив абсцисс или ординат функции}
                  LengthRPix,       {длина окна графика по оси X или Y (в пикселях)}
                  NR: integer;      {число линий сетки по оси X или Y}
                var MinR, MaxR,     {мин. и макс. значения абсцисс или ординат функции}
                  DelR: real;       {шаг сетки по оси X или Y}
                var MaschtabR: integer);  {масштаб по оси X или Y}
```

```
var
  Lr: real; {диапазон графика по оси X или Y}

  {Округление шага отсчетов Delta (оцифровки) по оси X или Y}
  procedure RoundDelta(var Delta: real); {шаг отсчетов по оси X или Y}
  const
    NMas = 7; {количество вариантов отсчетов линий сетки в диапазоне 1..10}
    {варианты отсчетов линий сетки}
    Mas: array[1..NMas] of real = (1, 2, 2.5, 4, 5, 8, 10);
  var

    i,                  {индекс}
    iMin : integer;     {индекс Delta с минимальной ошибкой округления}
    koef,               {масштабный коэффициент Delta}
    Del,                {шаг отсчетов по оси}
    MinDelta: real;     {Delta с минимальной ошибкой округления}

  begin
    {Приводим Delta к диапазону 1..10}
    Koef := 1;  {начальное значение масштабного коэффициента Delta}
    while (Delta < Mas[1]) or (Delta > Mas[NMas]) do
      begin
        if Delta < Mas[1]
          then begin
                 Delta := 10 * Delta;
                 Koef := 10 * Koef
               end;
        if Delta > Mas[NMas]
          then begin
                 Delta := 0.1 * Delta;
                 Koef := 0.1 * Koef
               end
      end;

    {Округляем до ближайшего допустимого варианта оцифровки}
    MinDelta := 1e38;
    for i := 1 to NMas do
      begin
        {абсолютная ошибка при использовании i-го типового варианта оцифровки}
        Del := abs(Delta - Mas[i]);
        if Del < MinDelta

          then begin
                 MinDelta := Del;
                 iMin := i
               end
      end;
    Delta := Mas[iMin] / koef {шаг сетки по оси X или Y с миним. ошибкой округления}

  end; {RoundDelta}
```

```
begin

  {Поиск мин. MinR и макс. MaxR элементов в массиве абсцисс или ординат}
  MinMax(N, R, MinR, MaxR);
  Lr := MaxR - MinR;      {диапазон графика по оси X или Y}
  DelR := Lr / (NR + 1); {приближенный шаг линий сетки по оси X или Y}
  RoundDelta(DelR);       {Округление шага линий сетки по оси X или Y}
  MaschtabR := round(LengthRPix / Lr)  {масштаб по оси X или Y}

end; {Mashtab}

procedure LinX(xi: LongInt); {проведение вертикальной оси через точку xi}
begin
  if (xi >= x1) and (xi <= x2) {если попадаем в окно,}
     then line(xi, y1, xi, y2)  {то проводим вертикальную ось}
end; {LinX}

procedure LinY(yi: LongInt); {проведение горизонтальной оси через точку yi}
begin
  if (yi >= y1) and (yi <= y2)  {если попадаем в окно,}
    then line(x1, yi, x2, yi)       {то проводим горизонтальную ось}
end; {LinY}

{Вывод значения оцифровки по оси X}
procedure ValX(x, y: integer;          {координаты выводимого числа}
           var x1R, x2R: LongInt;      {допустимые границы вывода числа по оси X}
           R: real;                    {выводимое число}
           WidthR,                     {число позиций, занимаемое выводимым числом}
           Decimals: integer);         {число выводимых значащих цифр}
begin
  if (x >= x1) and (x <= x2)
    then Value(x, y, {Форматирование  и вывод числа в точке x, y}
           R,             {выводимое число}
           x1R, x2R,      {допустимые границы вывода числа по оси X}
           WidthR,        {число позиций, занимаемое выводимым числом}
           Decimals,      {число выводимых значимых цифр}
           CenterText)  {центрируем число относительно x, y}
end; {ValX}

{Вывод значения оцифровки по оси Y}
procedure ValY(x, y: integer; {координаты выводимого числа}
           R: real;                    {выводимое число}
           var x1R, x2R: LongInt;      {допустимые границы вывода числа по оси X}
           WidthR,                     {число позиций, занимаемое выводимым числом R}
           Decimals: integer);         {число выводимых значимых цифр}
begin
  if (y >= y1) and (y <= y2)
    then Value(x, y, {Форматирование и вывод числа в точке x, y}
           R,             {выводимое число}
           x1R, x2R,      {допустимые границы вывода числа по оси X}
```

```
            WidthR,      {число позиций, занимаемое выводимым числом R}
            Decimals,    {число выводимых значимых цифр}
            RightText)   {прижимаем число вправо, к коорд. x, y}
end; {ValY}

{Формирование строки с диапазонами координат выводимого графика}
Function Diapazon(NameX, NameY: tStr10;      {надписи к осям X и Y}
                  MinX, MaxX,                {мин. и макс. знач. абсцисс функции}
                  MinY, MaxY: real): tStr80; {мин. и макс. знач. ординат функции}
var
  St, Numb: tStr80; {вспомогательные строки}
begin
  Str(MinX:0:4, Numb);
  St := NameX + ' [' + Numb;
  Str(MaxX:0:4, Numb);
  St := St + ' .. ' + Numb + '],   ';
  Str(MinY:0:4, Numb);
  St := St + NameY + ' [' + Numb;
  Str(MaxY:0:4, Numb);
  St := St + ' .. ' + Numb + ']';
  Diapazon := St
end; {Diapazon}

{Формирование окна графика}
procedure WindowGr(NameGraph: tStr80; {заголовок графика функции}
                   NameX, NameY: tStr10;      {надписи к осям X и Y}
                   MinX, MaxX,                {мин. и макс. знач. абсцисс функции}
                   MinY, MaxY: real;          {мин. и макс. знач. ординат функции}
                   MaschtabX, MaschtabY: integer; {масштабы по осям X и Y}
                   ColorBk,                   {цвет фона графика}
                   ColorNameGraph,            {цвет заголовка графика}
                   ColorNameXY: Byte);        {цвет границ окна графика}
begin
  SetBkColor(ColorBk); {Установим цвет фона экрана}

  SetColor(ColorNameGraph); {Установим цвет заголовка графика}
  SetTextJustify(CenterText, CenterText); {Текст отцентрировать относительно курсора}

  OutTextXY(GetMaxX div 2, 12, NameGraph); {Вывод заголовка окна графика}

  OutTextXY(GetMaxX div 2, 25, {Вывод диапазонов координат}
        Diapazon(NameX, NameY, {надписи к осям X и Y}
                 MinX, MaxX,   {мин. и макс. знач. абсцисс функции}
                 MinY, MaxY)); {мин. и макс. знач. ординат функции}

  {Определяем координаты графического окна:}
  {координаты верхнего левого угла окна:}
  x1 := (GetMaxX - LengthXPix) div 2;
  y1 := (GetMaxY - LengthYPix) div 2;
  {координаты нижнего правого угла окна:}
  x2 := x1 + LengthXPix;
```

```
  y2 := y1 + LengthYPix;

  {Определяем координаты начала осей координат графика}
  x0 :=  x1 + round(-MinX * MaschtabX);
  y0 :=  y1 + round(MaxY * MaschtabY);

  {Установим цвет границ окна графика и параметры линии окна}
  SetColor(ColorNameXY);
  {Установим стиль линии – непрерывная, сплошная, утроенной толщины}
  SetLineStyle(SolidLn, 0, ThickWidth);
  Rectangle(x1, y1, x2, y2) {выводим границы окна}

end; {WindowGr}

{Вывод на экран сетки графика}
procedure Setka(NameGraph: tStr80;        {заголовок графика функции}
                NameX, NameY: tStr10;     {надписи к осям X и Y}
                MinX, MaxX,               {мин. и макс. значения абсцисс функции}
                MinY, MaxY,               {мин. и макс. значения ординат функции}
                DelX, DelY: real;         {шаг отсчетов (сетки) по осям X и Y}
                MaschtabX, MaschtabY: integer; {масштабы по осям X и Y}
                ColorBk,                  {цвет фона графика}
                ColorNameGraph,           {цвет заголовка графика}
                ColorNameXY,              {цвет границ окна графика}
                ColorSetka,               {цвет сетки и оцифровки графика}
                ColorOsi: Byte);          {цвет осей графика}

{Проведение вертикальных линий сетки справа от оси Y пока находимся в пределах окна}
procedure LineXPlus(R: real {значение оцифровки первой линии сетки});
var
  i: integer;
begin
  {Определяем допустимые границы вывода чисел оцифровки сетки справа от оси 0}
  x1R := x0 + TextWidth('0');
  if x1R < x1
    then x1R := x1;
  x2R := x2;

  {Проводим вертикальные линии сетки справа от оси 0, пока находимся в пределах окна}
  while xGr <= x2 do
    begin
      {проводим очередную линию сетки справа от оси 0, если возможно}
      LinX(xGr);

      {Выводим оцифровку линии сетки, если возможно}
      ValX(xGr, y2 + 12, {координаты выводимого числа}
           x1R, x2R,     {допустимые границы вывода числа оцифровки}
           R,            {значение оцифровки первой линии сетки}
           0,            {число позиций, занимаемое выводимым числом}
           1);           {число выводимых значимых цифр}
      {находим очередную абсциссу сетки справа от оси 0}
```

```
    xGr := xGr + round(DelX * MaschtabX);
    R := R + DelX;
    {Уточняем допустимые границы вывода числа оцифровки}
    if x1R <> x2R
      then x1R := x2R + TextWidth(' ') div 2;
    x2R := GetMaxX
  end
end;

{Проведение вертикальных линий сетки слева от оси Y, пока находимся в пределах окна}
procedure LineXMinus(R: real {значение оцифровки первой линии сетки});
begin
  {Определяем допустимые границы вывода числа оцифровки сетки слева от оси 0}
  x1R := x1;
  x2R := x0 - TextWidth(' ') div 2;
  if x2R > x2
    then x2R := x2;
  {Проводим вертикальные линии сетки слева от оси 0, пока находимся в пределах окна}
  while xGr >= x1 do
    begin
      {проводим очередную линию сетки слева от оси 0, если возможно}
      LinX(xGr);

      {Выводим оцифровку линии сетки, если возможно}
      ValX(xGr, y2 + 12,        {координаты выводимой линии сетки}
          x1R, x2R,             {допустимые границы вывода числа по оси X}
          R,                    {выводимое число}
          0,                    {число позиций, занимаемое выводимым числом}
          1);                   {число выводимых значимых цифр}
      {находим очередную абсциссу сетки справа от оси 0}
      xGr := xGr - round(DelX * MaschtabX);
      R := R - DelX;
      {Уточняем допустимые границы вывода числа оцифровки}
      x2R := x1R - TextWidth(' ') div 2;
      x1R := 0
    end
end;

{Проведение горизонтальных линий сетки выше оси X пока находимся в пределах окна}
procedure LineYPlus;
begin
  while (y0 - yGr) >= y1 do  {пока находимся в пределах окна}
    begin
      {Проводим очередную горизонтальную линию сетки выше оси X, если возможно}
      LinY(y0 - yGr);
      {Определяем допустимые границы вывода числа оцифровки}
      if x1R <> x2R
        then x1R := 0;
      x2R := x2;
      {Выводим оцифровку линии сетки, если возможно}
      ValY(x1 - TextWidth('0'), y0 - yGr,       {координаты выводимого числа}
```

```
        YGr / MaschtabY,                {выводимое число}
        x1R, x2R,                       {допустимые границы вывода числа по оси X}
        0,                              {число позиций, занимаемое выводимым числом}
        1);                             {число выводимых значимых цифр}
    {находим очередную ординату сетки ниже оси 0}
    yGr := yGr + round(DelY * MaschtabY)
  end
end;

{Проведение горизонтальных линий сетки ниже оси X пока находимся в пределах окна}
procedure LineYMinus;
begin
 while (y0 + YGr) <= y2 do {пока находимся в пределах окна}
  begin
    {проводим очередную линию сетки ниже оси Y, если возможно}
    LinY(y0 + yGr);

    x1R := 0;  x2R := x2; {допустимые границы вывода числа оцифровки}
    {Выводим оцифровку линии сетки, если возможно}
    ValY(x1 - TextWidth('0'), y0 + yGr,     {координаты выводимого числа}
       - YGr / MaschtabY,                   {выводимое число оцифровки оси}
       x1R, x2R,                            {допустимые границы вывода числа по оси X}
       0,                                   {число позиций, занимаемое выводимым числом}
       1);                                  {число выводимых значимых цифр}
    {находим очередную ординату сетки ниже оси 0}
    yGr := yGr + round(DelY * MaschtabY)
  end
end;

begin

 {Формирование окна графика}
 WindowGr(NameGraph,            {заголовок графика функции}
       NameX, NameY,            {надписи к осям X и Y}
       MinX, MaxX,              {мин. и макс. знач. абсцисс функции}
       MinY, MaxY,              {мин. и макс. знач. ординат функции}
       MaschtabX, MaschtabY,    {масштабы по осям X и Y}
       ColorBk,                 {цвет фона графика}
       ColorNameGraph,          {цвет заголовка графика}
       ColorNameXY);            {цвет границ окна графика}

 SetColor(ColorSetka); {Установим цвет сетки графика}
 SetLineStyle(DottedLn, 0, NormWidth);

 {Формируем вертикальные линии сетки и их оцифровку}
 if (x0 >= x1) and (x0 <= x2)
  then begin {ось Y внутри окна графика}
       {абсцисса первой линии сетки справа от оси X}
       xGr := x0 + round(DelX * MaschtabX);
       {Проведение вертикальных линий сетки справа от оси X}
       LineXPlus(DelX);
```

```
        {абсцисса первой линии сетки слева от оси X}
        xGr := x0 - round(DelX * MaschtabX);
        {Проведение вертикальных линий сетки слева от оси X}
        LineXMinus(-DelX)
      end
  else if x0 > x2
        then begin {ось Y справа от окна графика}
             {абсцисса первой вертикальной линии сетки в окне}
             xGr := x0 - round(round(abs(MaxX / DelX)) * DelX * MaschtabX);
             {Проведение вертикальных линий сетки слева от оси X}
             LineXMinus((xGr - x0) /MaschtabX)
            end
        else begin {ось Y слева от окна графика}
             {абсцисса первой вертикальной линии сетки в окне}
             xGr := round(trunc(abs(MinX / DelX)) * DelX * MaschtabX) + x0;
             {Проведение вертикальных линий сетки справа от оси X}
             LineXPlus((xGr - x0) / MaschtabX)
            end;

{Формируем горизонтальные линии сетки и их оцифровку}
if (y0 >= y1) and (y0 <= y2)
  then begin {ось X внутри окна графика}
        yGr := round(DelY * MaschtabY); {ордината первой линии}
        LineYPlus; {Проведение горизонтальных линий сетки выше оси X}
        yGr := round(DelY * MaschtabY); {ордината первой линии}
        LineYMinus {Проведение горизонтальных линий сетки ниже оси X}
      end
  else if y0 > y2
        then begin {ось X ниже окна графика}
             yGr := round(round(abs(MinY / DelY)) * DelY * MaschtabY);
             LineYPlus {Проведение горизонтальных линий сетки выше оси X}
            end
        else begin {ось X выше окна графика}
             yGr := round(round(abs(MaxY / DelY)) * DelY * MaschtabY);
             LineYMinus {Проведение горизонтальных линий сетки ниже оси X}
            end;

{Вывод осей координат графика}

{Установим цвет осей графика и параметры линии осей}
SetColor(ColorOsi);
SetLineStyle(SolidLn, 0, NormWidth);
LinX(x0);   LinY(y0);  {проводим оси, проходящие через 0,0 (если возможно)}

{Оцифровка осей координат}
SetTextJustify(CenterText, CenterText); {Текст отцентрировать относительно курсора}
if (x0 >= x1) and (x0 <= x2) {выводим оцифровку оси X (если возможно)}
  then OutTextXY(x0, y2+12, '0');
if (y0 >= y1) and (y0 <= y2) {выводим оцифровку оси Y (если возможно)}
  then OutTextXY(x1 - 12, y0, '0');
```

```
  {Вывод наименования осей}
  SetColor(ColorNameXY); {цвет наименования осей}
  SetTextJustify(RightText, CenterText);          {Текст прижать вправо к курсору}
  OutTextXY(x1 - TextWidth('0'), y1, NameY);      {Вывод надписи к оси Y}

  SetTextJustify(LeftText, CenterText);           {Текст прижать влево к курсору}
  OutTextXY(x2 + TextWidth('0'), y2, NameX)       {Вывод надписи к оси X}

end; {Setka}

{Вывод на экран графика функции, выполненного точками}
procedure GraphFun(VarGr: byte; {VarGr - вариант вывода графика:
                                 1 - точками,
                                 2 - линией,
                                 3 - кружками,
                                 4 - прямоугольниками,
                                 5 - гистограма}
                   MaschtabX,        {масштаб по оси X}
                   MaschtabY,        {масштаб по оси Y}
                   N: Integer;       {число расчетных точек функции}
                   X, Y: tMassiv;    {массивы абсцисс и ординат функции}
                   ColorGraph: byte);  {цвет графика}

{Преобразование абсциссы точки графика x в абсциссу графического окна}
function GrX(x: real): integer;
var
  gX: integer;
begin
  gX := x0 + round(x * MaschtabX);
  if gX > x2
    then gX := x2;
  if gX < x1
    then gX := x1;
  GrX := gX
end;

{Преобразование ординаты точки графика y в ординату графического окна}
function GrY(y: real): integer;
var
  gY: integer;
begin
  gY := y0 - round(y * MaschtabY);
  if gY > y2
    then gY := y2;
  if gY < y1
    then gY := y1;
  GrY := gY
end;

var
  i: integer;
```

```
begin
  SetColor(ColorGraph);                          {Устанавливаем цвет выводимой функции}
  SetLineStyle(SolidLn, 0, NormWidth);           {линия сплошная, нормальной толщины}
  MoveTo(GrX(X[1]), GrY(Y[1]));          {перемещаем курсор в первую точку графика}
  PutPixel(GrX(X[1]), GrY(Y[1]), ColorGraph);    {выводим первую точку графика}

  for i := 2 to N do  {Цикл формирования кривой графика}
   begin
    xGr := GrX(X[i]); yGr := GrY(Y[i]);
    case VarGr of {определяем вид кривой графика и строим его очередную точку}
     1: PutPixel(xGr, yGr, ColorGraph);          {точками}
     2: LineTo(xGr, yGr);                        {отрезками линиями}
     3: Circle(xGr, yGr, 2);                     {кружками с радиусом 2}
     4: Rectangle(xGr-2, yGr-2, xGr+2, yGr+2);   {квадратами со стороной 4}
     5: begin                          {закрашенными прямоугольниками - гистограмма}
          if (yGr >= y1)  and (yGr <= y2)
           then yGr := y0
           else begin
                 if yGr >= y1
                  then yGr := y1;
                 if yGr <= y2
                  then yGr := y2
                end;
         Bar(GrX(X[i-1]), GrY(Y[i-1]), xGr, yGr)
        end
   end
  end
end; {GraphFun}

{Форматирование  и вывод числа в заданную точку экрана в графическом режиме}
procedure Value(x, y: integer; {координаты выводимого числа}
            R: real;       {выводимое число}
            var x1R, x2R: LongInt; {допустимые границы вывода числа по оси X}
            WidthR,          {число позиций, занимаемое выводимым числом}
            Decimals: integer; {число выводимых значимых цифр}
            Horizont: word); {выравнивание: LeftText, CenterText, RightText}
var
  St: string[15];     {строка для преобразования числа в текст}
  WidthRPix: integer; {ширина в пикселах числа, выводимого по оси X}
  OldStyle : TextSettingsType; {предыдущий стиль текста}
begin
  GetTextSettings(OldStyle); {Возвращает текущий шрифт, направление, размер и
                              выравнивание текста, установленные процедурами SetTextStyle и
                              SetTextJustify}
  SetTextStyle(SmallFont, HorizDir, 4); {Устанавливаем малый векторный шрифт
LITT.CHR}
  Str(R: WidthR: Decimals, St);                {преобразуем число R в строку St}
  WidthRPix := TextWidth(St) div 2;   {половина ширины числа (в пикселях), выводимого по
                                       оси X}
  if (x - WidthRPix >= x1R) and (x + WidthRPix <= x2R)
   then begin
```

```
        {текст выровнять относительно курсора}
        SetTextJustify(Horizont, CenterText);
        OutTextXY(x, y, St); {вывод числа}
        {возвращаем уточненные границы вывода числа}
        x1R := x - WidthRPix; x2R := x + WidthRPix
      end
    else
      if x > x0
       then x2R := x1R
      else
       x1R := x2R;
  {Восстанавливаем старый стиль}
  With OldStyle do
   begin
    SetTextJustify(Horiz, Vert);
    SetTextStyle(Font, Direction, CharSize)
   end
end; {Value}

procedure Pause; {временная остановка до нажатия <<Enter>>}
begin
  SetColor(Yellow);
  SetTextJustify(CenterText, CenterText);
  OutTextXY(GetMaxX div 2, GetMaxY -10, 'Для завершения работы нажмите <Enter>...');
  Readln;
end;

begin     {Секция инициализации модуля (пустая)}

end.      {Конец модуля Unit12_2}
```

7 Индивидуальные задания

7.1 Требования к программе и рекомендации по ее разработке

При выполнении задания в качестве заготовки программы разумно использовать рассмотренный выше учебный пример. Для этого следует скопировать файлы ***Prim12_2.pas*** и ***Unit12_2.pas*** в свой каталог, переименовать их и взять затем за основу разрабатываемой программы. Практически, для большинства вариантов задач изменения в модуле ***Unit12_2.pas*** могут не потребоваться.

Тестовый пример формируется самостоятельно. Он должен позволять оценить работоспособность программы.

При составлении программы по индивидуальному заданию предусмотреть:

- ✓ использование структурного и модульного подхода;
- ✓ простейший диалог типа «запрос-ответ» при вводе данных;
- ✓ многократный ввод данных при исполнении программы, т.е. повторный счет[8] при других диапазонах изменения аргумента X, числе линий сетки по осям X и Y и т.п.;

[8] Вместо констант, используемых при обращении к процедурам, и напрямую задающим соответствующие параметры, разумно ввести переменные и вводить их значения в режиме «запрос-ответ».

- ✓ вывод результатов в удобном для пользователя виде;
- ✓ освоить методику поиска причин и исправления ошибок, а также трассировки программы «по шагам»[9] по тестовому примеру.
- ✓ подпрограммы, составленные при выполнении задания, должны быть оформлены в виде пользовательского модуля.

7.2 Варианты заданий

Определить графически приближенные значения корней[10] нелинейных уравнений, заданных по своему варианту. Обратите внимание, что заданная функция возможно существует лишь в ограниченном диапазоне, когда у функций ***Ln(x)*** и ***Sqrt(x)*** аргументы ***x > 0***, а для ***exp(x)*** не слишком велики, чтобы не возникло переполнение разрядной сетки ПК и т.п.

1. $y = f(x) = \ln(x) + (x+1)^3 = 0.$
2. $y = f(x) = \sqrt{(x+1)} - 1 / x = 0.$
3. $y = f(x) = 2*x - \cos(x) = 0.$
4. $y = f(x) = (2-x)*e^x = 0.$
5. $y = f(x) = x^2 + 4*\sin(x+1) = 0.$
6. $y = f(x) = 3*x + \sin(x) - (x+1)^3 = 0.$
7. $y = f(x) = 5*x - \ln(x) + (x-1)^2 = 0.$
8. $y = f(x) = 2*\sin(0.5 + x) + 3*x^2 = 0.$
9. $y = f(x) = x - \cos(x) - 1 = 0.$
10. $y = f(x) = x^2 – 2*\sin(x) = 0.$
11. $y = f(x) = 2*\sin(x-0.8) + x - 1.5 = 0.$
12. $y = f(x) = \ln(x) - (3-x)*e^x = 0.$
13. $y = f(x) = 3*\sin(x-2) + (x-1)^2 = 0.$
14. $y = f(x) = 5*\cos(0.6*x - 2) + (x-1)^3 = 0.$
15. $y = f(x) = -3*\sin(2*x) + \ln(x-1) = 0.$
16. $y = f(x) = -2*\cos(0.4*x-1) + e^{x+1} = 0.$
17. $y = f(x) = \ln(3*x) + (x-0.6)^2 = 0.$
18. $y = f(x) = \ln(2*x) + (x+2)^3 = 0.$
19. $y = f(x) = \cos(2*x-1) + \ln(x+1)^2 = 0.$
20. $y = f(x) = 3*\sin(x-3) - (x-2)^2 = 0.$
21. $y = f(x) = 5*\cos(x) + (x-1)^2 = 0.$
22. $y = f(x) = -3*\ln(x) + 0.9*(x+1)^3 = 0.$
23. $y = f(x) = 5*\ln(x) - (x-1)^3 = 0.$
24. $y = f(x) = 4*\cos(x-3) + (x-1)^3 = 0.$
25. $y = f(x) = 2*\sin(x+2) + (x+1)^3 = 0.$

Список рекомендуемой литературы

1. Д. Ван Тассел. Стиль, разработка, эффективность, отладка и испытание программ: Пер. с англ. – М.: Мир, 1985. – 332 с.
2. Н. Вирт. Алгоритмы и структуры данных. : Пер. с англ. – М.: Мир, 1989. – 360 с.
3. Основы информатики. Учеб. Пособие / Аладьев В.З., Хунт Ю.Я., Шишаков М.Л. - М.: Информационно-издательский дом "Филинъ", 1998. - 496 с.
4. Марченко А.И., Марченко Л.А. Программирование в среде Borland Pascal 7.0. – К.: ЮНИОР, 1998. – 480 с.

[9] Используйте функциональные клавиши F7 или F8.

[10] Корень уравнения обращает функцию в 0 при пересечении оси ***X***.

5. Зуев Е.А. Программирование на языке Турбо Паскаль 6.0, 7.0. - М: Веста, Радио и связь, 1993. - 384 с.
6. Епанешников А., Епанешников В. Программирование в среде Turbo Pascal 7.0. - М.: "ДИАЛОГ-МИФИ", 1993. - 288 с.
7. Фаронов В.В. Turbo Pascal 7.0. Начальный курс. Учебное пособие.- М.: "НОЛИДЖ", 2001. - 576 с.
8. Фаронов В.В. Turbo Pascal 7.0. Практика программирования. Учебное пособие.- М.: "НОЛИДЖ", 1998. - 432 с.
9. ОС ТУСУР 6.1-97. Работы студенческие учебные и выпускные квалификационные. - Томск: ТУСУР, 1999.- 10 с.