ГОСУДАРСТВЕННЫЙ КОМИТЕТ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПО ВЫСШЕМУ ОБРАЗОВАНИЮ

САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С. П. КОРОЛЕВА

ГРАФИКА В СИСТЕМЕ ТУРБО ПАСКАЛЬ

Методические указания к лабораторным работам

САМАРА 1995

Составитель Е. А. Логвинова

УДК 681.3.06

Графика в системе Турбо Паскаль: Метод. указания к лаб. работам /Самар. аэрокосм. ун-т; Сост. Е. А. Логвинова. Самара, 1995. 20 с.

Содержатся указания по выполнению лабораторных работ на популярном алгоритмическом языке программирования Турбо Паскаль с использованием графики. Подробно рассматриваются подпрограммы модуля GRAPH (библиотека графических подпрограмм универсального назначения), даны многочисленные примеры их использования. Показано, как инициализировать графический режим и выйти из него, перечислены характерные ошибки инициализации.

Предназначены для выполнения лабораторных работ по курсу «ЭВМ и программирование», «Вычислительная техника и программирование», «Информатика» для студентов всех специальностей. Составлены на кафедре «Программное обеспечение вычислительных систем».

Печатаются по решению редакционно-издательского совета Самарского государственного аэрокосмического университета им. академика С. П. Королева

Рецензент А. О. Новиков

Цель работы: Использование графики в прикладных программах на алгоритмическом языке программирования Турбо Паскаль.

МОДУЛЬ GRAPH

Все системные библиотеки Турбо Паскаля реализованы как модули, и чтобы воспользоваться, например, библиотекой подпрограмм модуля GRAPH, нужно указать директиву Uses GRAPH в начале программы. Модуль CRAPH содержит типы, константы, переменные и подпрограммы, позволяющие программисту создавать изображения с использованием широкого набора графических адаптеров (CGA, MCGA, EGA, VGA, HERCULES, AT&400, 320PC, IBM8514).

При работе с этими адаптерами весь экран разбивается на определенные «точки» — пиксели, которые можно закрасить в тот или иной цвет. Каждый пиксель имеет две координаты: X и Y. Координата X увеличивается по горизонтали слева направо, начиная от нуля. Координата Y увеличивается по вертикали сверху вниз. Таким образом, левый верхний пиксель имеет координаты (0,0). Количество пикселей зависит от типа графического адаптера и режима его работы.

Чтобы запустить программу, использующую процедуры модуля GRAPH, необходимо, чтобы в рабочем каталоге находились соответствующие графические драйверы (файлы с расширением .BGI), которые обеспечивают взаимодействие программ с графическими устройствами. При использовании в программе штриховых шрифтов необходимо, чтобы в рабочем каталоге находились файлы шрифтов (с расширением .CHR).

ИНИЦИАЛИЗАЦИЯ И ЗАКРЫТИЕ ГРАФИЧЕСКОГО РЕЖИМА

Для инициализации графического режима используется процедура InitGraph, которая определяет тип адаптера и подключает соответствующий драйвер, который и устанавливает графический режим:

Процедура InitGraph (var grDriver: Integer, {тип адаптера}
var grMode: Integer, {режим графики}
DriverPath: String); {путь к драйверу}

Если grDriver = Detect (или 0), то система включает режим автоопределения. Параметры grDriver и grMode примут автоматически выбранные значения. При автоопределении включается режим с максимальным разрешением.

Если же параметр grDriver содержит номер адаптера, то в этом случае параметр grMode должен иметь значение режима, допустимого для этого адаптера. Для определения этих параметров используется функция DetectGraph.

DriverPath указывает путь в каталог, содержащий файлы с необходимыми драйверами. Если в него передается пустая строка '', то драйверы должны находиться в текущем каталоге.

Процедура DetectGraph (Var grDriver, grMode: Integer);
где grDriver — тип графического адаптера;
grMode — режим работы.

DetectGraph определяет тип графического адаптера и допустимый режим его работы. Функция DetectGraph может быть вызвана перед процедурой инициализации InitGraph.

В модуле GRAPH реализованы следующие способы проверки инициализации при помощи функций:

Функция GraphResult: Integer;

GraphResult возвращает код результата вызова процедуры инициализации InitGraph (существуют и другие функции и процедуры, коды результатов которых возвращает эта функция).

Функция GraphErrorMsg (ErrorCode: Integer): String;

GraphErrorMsg преобразует результат вызова функции GraphResult в сообщение, которое можно вывести на экран процедурой Writeln:

```
Writeln ('Результат инициализации графики:',
                    GraphErrorMsg(ErrorCode));
```

Наиболее распространенные коды ошибок при инициализации

0 No error — ошибок нет;

— 1 — (BGI) graphics not installed — графика не инициализирована;

— 3 — Device driver file not found ( ) — BGI файла нет в указанном каталоге.

Пример инициализации графического режима (BGI — файлы находятся на диске D, каталог TP, подкаталог BGI):

```
Uses Graph;
Procedure INIT; {Процедура ИНИЦИАЛИЗАЦИИ}
  var grDriver: Integer; {графический адаптер}
      grMode: Integer; {графический режим}
      ErrorCode: Integer; {код ошибки}
  Begin
      grDriver:=Detect; {режим автоопределения}
      InitGraph(grDriver, grMode,' D: \TP \ BGI '); {инициализация}
      ErrorCode: = GraphResult; {результат инициализации}
      if ErrorCode<>grOk then {если не успешно, то...}
        begin
          writeln (' Ошибка графики:', GraphErrorMsg
          (ErrorCode));
          Halt(1), {прекращение работы программы}
        end;
  End;                          {конец процедуры INIT}

  BEGIN   {начало основной программы}
      Init;     {вызов процедуры инициализации}
      {... работа в графическом режиме...}
      GloseGraph; {выйти из графического режима}
  END.
```

Процедура CloseGraph завершает работу в графическом режиме и осуществляет переход в текстовый режим. Возврат в графический режим возможен только через повторную инициализацию.

В процессе работы в графическом режиме можно осуществлять переход в текстовый режим и обратно в графический, минуя повторную инициализацию. Для этого используются следующие процедуры и функции:

Функция GetGraphMode: Integer; — возвращает номер текущего графического режима;

Процедура RestoreCrtMode; — возвращает систему в текстовый режим;

Процедура SetGraphMode(Mode: Integer); — восстанавливает графический режим без повторной инициализации и очищает экран

Параметр Mode (режим) должен задавать для текущего драйвера устройства допустимый графический режим. Данная процедура используется для выбора графического режима, отличного от того, который по умолчанию устанавливается процедурой InitGraph.

Пример:

```
Uses Graph;
  Begin
     INIT;
     OutText(' для выхода из режима графики нажмите
     <ENTER>');
     ReadIn;
     RestoreCRTMode;
     WriteIn (' теперь Вы в текстовом режиме ');
     WriteIn (' для перехода в режим графики нажмите
     <ENTER>');
     ReadIn;
     SetGraphMode(GetGraphMode);
     OutText(' вы снова в графическом режиме.');
     OutText(' для выхода нажмите <ENTER>');
     ReadIn;
     CloseGraph;
  End
```

ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ ГРАФИЧЕСКОГО РЕЖИМА И УСТАНОВКА ПАРАМЕТРОВ ИЗОБРАЖЕНИЙ

Процедура ClearDevice;

ClearDevice очищает графический экран, закрашивает его в цвет фона, устанавливает указатель текущей позиции в точку с координатами (0,0). Цвет фона задается процедурой SetBkColor.

Процедура ClearViewPort;

ClearViewPort очищает выделенное окно графического экрана, закрашивает его в цвет фона, устанавливает указатель текущей позиции в левый верхний угол окна с координатами (0,0).

Известно, что отношение высоты экрана к ширине не равно отношению его разрешающей способности по вертикали к разрешающей способности по горизонтали. Для учета этого неравенства существует процедура, которая определяет коэффициент сжатия изображения:

Процедура GetAspectRatio (var Xasp, Yasp: Word);

GetAspectRatio в переменных Xasp и Yasp возвращает значения, отношение которых (Xasp/Yasp) соответствует коэффициенту сжатия изображения, т. е. Xasp и Yasp — переменные, в которых процедура записывает число точек по горизонтали и вертикали.

Процедура SetAspectRatio (Xasp, Yasp: Word);

SetAspectRatio меняет относительный масштаб изображения по горизонтали и вертикали, где Xasp, Yasp — задаваемые масштабы по горизонтали и вертикали.

Процедура SetViewPort (X1, Y1, X2, Y2: Integer; Clip: Boolean);

SetViewPort задает размеры окна для вывода графической информации и перемещает указатель текущей позиции в левый верхний угол с координатами (0,0),

где X1, Y1 — координаты левого верхнего угла окна;

X2, Y2 — координаты нижнего правого угла окна (0 <= X1 < X2 и 0 <= Y1 < Y2);

Clip — ограничитель фигур. Если Clip = True, то все построения проводятся только в пределах окна, в противном случае части фигур могут выходить за пределы окна.

ПРОЦЕДУРЫ И ФУНКЦИИ УПРАВЛЕНИЯ ЦВЕТОМ

Процедура SetBkColor (Color: Word); — задает текущий цвет фона экрана;

Функция GetBkColor: Word; — возвращает номер текущего цвета фона экрана;

Процедура SetColor (Color: Word); — устанавливает текущий цвет;

Функция GetColor: Word; — возвращает номер текущего цвета;

Функция GetMaxColor: Word; — возвращает максимальный номер цвета, воспринимаемого данным адаптером в текущем графическом режиме.

В табл. 1 приведены коды цветов для подпрограмм управления цветом:

Таблица 1

Коды цветов

Имя	Значение	Назначение
Black	0	Черный
Blue	1	Синий
Green	2	Зеленый
Cyan	3	Голубой
Red	4	Красный
Magenta	5	Фиолетовый
Brown	6	Коричневый
Light Gray	7	Светло-серый
DarkGray	8	Темно-серый
LightBlue	9	Светло-синий
LightGreen	10	Светло-зеленый
LightCyan	11	Светло-голубой
LightRed	12	Розовый
LightMagenta	13	Светло-фиолетовый
Yellow	14	Желтый
White	15	Белый

ФУНКЦИИ РАБОТЫ С ТОЧКАМИ

Функция GetMaxX: Integer; — получение максимального значения координаты X;

Функция GetMaxY: Integer; — получение максимального значения координаты Y;

Функция GetX: Integer; — получение координаты X текущей позиции на экране;

Функция GetY: Integer; — получение координаты Y текущей позиции на экране.

Пример:

```
Uses Graph;
  Begin
     INIT;                    {нарисовать полную рамку экрана}
     Rectangle (0, 0, GetMaxX, GetMaxY);
     ReadIn;
     CloseGraph;
  End.
```

ПРОЦЕДУРЫ РАБОТЫ С ЛИНИЯМИ

В Турбо Паскале можно управлять стилем линий: задавать толщину и тип. Для этого определены следующие тип и константы стилей изображаемых линий:

```
Тип LineSettingsType = record
                         LineStyle: Word; {стиль (тип)}
                         Pattern: Word; {шаблон}
                         Thickness: Word; {толщина}
                       end;
```

```
Процедура SetLinesStyle (LineStile: Word; Pattern: Word;
                                          Thickness: Word);
```

SetLineStyle задает текущие параметры линии (стиль, шаблон и толщину),

где LineStyle — стиль линии (табл. 2);

Таблица 2

Типы линий для процедур SetLineStyle, GetLineSettings

Имя	Значение	Назначение
SolidLn	0	Сплошная
DottedLn	1	Пунктирная
CenterLn	2	Штрихпунктирная
DashedLn	3	Штриховая
UserBitLn	4	Заданная пользователем

Pattern — шаблон линии — задается только в случае, если LineStyle = UserBitLn; представляется в виде двухбайтового числа; каждый бит которого = 1, если очередной пиксель следует высветить, 0 — в противном случае;

Thickness — толщина линии (табл. 3).

Таблица 3

Толщины линий для процедур SetLineStyle, GetLineSettings

Имя	Значение	Назначение
NormWidth	1	Нормальная
ThickWidth	3	Толстая

Процедура GetLineSettings (Var LineInfo: LineSettingType);

GetLineSettings возвращает текущие параметры линии (стиль, шаблон и толщину),

где LineInfo — переменная, в которую записываются текущие параметры линии (см. табл. 2, 3).

Процедура SetWriteMode (Mode: Integer);

SetWriteMode устанавливает режим вычерчивания линий

Если Mode = 0 — линия пересекает существующую картинку;

Mode = 1 — линия не проходит по местам, «занятым» существующими пикселями.

Пример:

```
Uses Graph;
  Var OldStyle: LineSettingsType;
  Begin
      INIT;
      Line (0, 0, 100, 0);          {тонкая, сплошная линия}
      GetLineSettings(OldStyle);    {запомнить старый стиль}
      SetLineStyle(DashedLn,0, ThickWidth);
      Line (0, 10, 100, 10);        {жирная, пунктирная линия}
      with OldStyle do   {восстановить старый стиль}
      SetLineStyle(LineStyle, Pattern, Thickness);
      Line(0, 20, 100, 20);     {тонкая, сплошная линия}
      ReadLn;
      CloseGraph;
  End.
```

ПОСТРОЕНИЕ ИЗОБРАЖЕНИЙ

Процедура Line (X1, Y1, X2, Y2: Integer);

Line рисует на экране линию (отрезок),
где X1, Y1 — координаты начальной точки;
X2, Y2 — координаты конечной точки;

Процедура Circle (X, Y: Integer; Radius: Word);

Circle рисует на экране окружность,
где X, Y — координаты центра окружности;
Radius — ее радиус.

Если коэффициент сжатия изображения будет соответствовать принятому BGI драйвером, то результатом работы будет окружность, иначе получится эллипс.

Процедура Arc(X, Y: Integer; StartAngle, EndAngle,
Radius: Word);

Arc рисует дугу радиуса Radius из центра с координатами (X, Y) от угла StartAngle до угла EndAngle (диапазон 0..360).

Процедура Ellipse (X, Y: Integer; StartAngle,
EndAngle, XRadius, YRadius: Word);

Ellipse рисует эллиптическую дугу из центра с координатами (X, Y) от угла StartAngle до угла EndAngle (диапазон 0 ... 360). XRadius и YRadius — размеры горизонтальной и вертикальной полуосей соответственно. Оси эллипса могут быть только параллельны осям X и Y. Для изображения полного эллипса надо задавать углы 0 и 360 градусов. Значение коэффициента сжатия изображения не влияет на построение.

Процедура Rectangle (X1, Y1, X2, Y2: Integer);

Rectangle рисует прямоугольник. Точка с координатами (X1, Y1) определяет верхний левый угол, а точка с координатами (X2, Y2) — нижний правый угол (0 <= X1 < X2 <= GetMaxX, 0 <= Y1 < Y2 <= GetMaxY).

Процедура PutPixel (X, Y: Integer; Pixel: Word);

PutPixel закрашивает пиксель с заданными координатами (X, Y) в цвет Pixel.

Процедура LineTo (X, Y: Integer);

LineTo проводит линию текущего цвета из текущей позиции в точку с заданными координатами (X, Y).

Процедура LineRel (Dx, Dy: Integer);

LineRel проводит линию текущего цвета из текущей позиции в точку, сдвинутую относительно исходной позиции на величины Dx и Dy. Dx и Dy — смещение координат точки конца линии по отношению к исходным координатам.

Процедура MoveRel (Dx, Dy: Integer);

MoveRel перемещает указатель координат на заданное расстояние (Dx, Dy) по отношению к его предыдущему положению (точка на экране не высвечивается).

Процедура DrawPoly (NumPoints: Word; Var PolyPoints);

DrawPoly строит контур многоугольника линией с заданными параметрами и цветом,

где NumPoints — число вершин многоугольника,

PolyPoints — переменная без типа, содержащая NumPoints + 1 пар координат вершин многоугольника (координаты должны быть целого типа и перечисляться в той же последовательности, как они идут по контуру, причем первая вершина должна быть повторена в конце перечисления).

ЗАКРАШИВАНИЕ ИЗОБРАЖЕНИЙ

В Турбо Паскале можно управлять образцом закраски изображений и ее цветом. Для этого определены следующий тип и константы:

```
Тип FillSettingsType = record
Pattern: Word; { образец закраски}
Color: Word, {цвет закраски}
end;
```

Процедура SetFillStyle (Pattern: Word; Color: Word);

SetFillStyle задает стандартный орнамент и цвет заполнения фигур,

где Pattern — определяет вид шаблона заполнения (табл. 4);

Color — цвет заполнения.

Процедура GetFillSettings (Var FillInfo: FillSettingsType);

GetFillSettings возвращает текущие орнамент и цвет заполнения, установленные процедурами SetFillStyle или SetFillPattern.

Таблица 4

Константы орнамента заполнения для процедур SetFillStyle и GetFillSettings

Имя	Значение	Назначение
EmptyFill	0	Заполнение цветом фона
SolidFill	1	Однородное заполнение цветом
LineFill	2	Заполнение — — —
LtSlashFill	3	Заполнение ///
SlashFill	4	Заполнение /// толстыми линиями
BkSlashFill	5	Заполнение \\\ толстыми линиями
LtBkSlashFill	6	Заполнение \\\
HatchFill	7	Заполнение клеткой
XHatchFill	8	Заполнение косой клеткой
InterLeaveFill	9	Заполнение частой сеткой
WideDotFill	10	Заполнение редкими точками
CloseDotFill	11	Заполнение частыми точками
UserFill	12	Заполнение определяется пользователем

Процедура FillPoly (NumPoints: Word; Var PolyPoints);

FillPoly строит многоугольник, закрашенный текущим орнаментом и цветом заполнения. Параметры определяются так же как в DrawPoly.

Процедура FillEllipse (X, Y: Integer; XRadius, YRadius: Word);

FillEllipse строит эллипс, закрашенный текущим орнаментом и цветом заполнения. Параметры определяются так же, как в Ellipse.

Пример:

```
Uses Graph;
  Begin
     INIT;
     SetFillStyle (3, 9); {тип: ///, ярко – голубой}
     FillEllipse (GetMaxX div 2, GetMaxY div 2, 90, 30);
     ReadIn;
     CloseGraph;
  End.
```

Процедура FloodFill (X, Y: Integer; Border: Word);

где X, Y — координаты любой точки внутри замкнутой области; Border — цвет линии, до которой производится закрашивание. FloodFill закрашивает область, ограниченную непрерывной линией, текущим орнаментом и цветом заполнения.

Пример:

```
Uses Graph;
  Begin
     INIT;
     SetColor (Green);
     Rectangle (150, 140, 280, 340); {зеленый прямоугольник}
     SetFillStyle (11, LightRed); { тип: :::, ярко – красный}
     FloodFill (160, 150, Green);  {закрасить}
     ReadIn;
     CloseGraph;
  End.
```

Процедура PieSlice (X, Y: Integer; StartAngle, EndAngle, Radius: Word);

PieSlice рисует и заполняет орнаментом сектор круга.

Процедура Bar (X1, Y1, X2, Y2: Integer);

Bar строит прямоугольник, закрашенный текущим орнаментом и цветом заполнения. (X1, Y1) — координаты левого верхнего угла прямоугольника, а (X2, Y2) — координаты правого нижнего угла.

Процедура Bar3D (X1, Y1, X2, Y2: Integer; Depth: Word; Top: Boolean);

Bar3D строит параллелепипед с закрашенной текущим орнаментом и цветом заполнения передней гранью,

где X1, Y1, X2, Y2 — координаты левого верхнего и правого нижнего углов передней грани;

Depth — ширина боковой грани (отсчитывается по горизонтали);

Top — признак включения верхней грани. Если Top = True, то верхняя грань вычерчивается, в противном случае верхняя грань не отображается (табл. 5). Это позволяет рисовать несколько параллелепипедов, расположенных один на другом (см. пример).

Таблица 5

Константы изображения верхней грани для Bar3D

Имя	Значение	Назначение
TopOn	True	Закрашивается верхняя грань
TopOff	False	Не закрашивается верхняя грань

Пример:

```
Uses Graph;
  Begin
     INIT;
     SetFillStyle (3, LightRed);
     Bar3D (10, 10, 50, 60, 10, TopOn); {вершина рисуется}
     SetFillStyle (HatchFill, 9);
     Bar3D (10, 60, 50, 110, 10, TopOff); {вершина не рисуется}
     Readln;
     CloseGraph;
  End.
```

Процедура Sector (X, Y: Integer, StartAngle, EndAngle, XRadius, YRadius: Word);

Sector строит сектор эллипса, закрашенный текущим орнаментом и цветом заполнения.

РАБОТА С ТЕКСТОМ В ГРАФИЧЕСКОМ РЕЖИМЕ

В комплектах пакета Турбо Паскаля имеются файлы с расширением .CHR — это набор штриховых шрифтов, которые могут быть использованы для вывода текста в графическом режиме.

Для обозначения шрифтов введены следующие константы (табл. 6).

Таблица 6

Константы шрифтов

Имя	Значение	Назначение
DefaultFont	0	Матричный шрифт 8x8 (по умолчанию)
TriplexFont	1	Полужирный
SmallFont	2	Светлый (тонкое начертание)
SansSerifFont	3	Книжная гарнитура
GothicFont	4	Готический шрифт

Активизация шрифта осуществляется следующей процедурой:

Процедура SetTextStyle (Font: Word, Direction: Word; CharSize: Word),

где Font — тип шрифта (табл. 6);
Direction — расположение текста (табл. 7);

Таблица 7

Направление (расположение) текста

Имя	Значение	Назначение
HorizDir	0	Горизонтальное расположение строк
VertDir	1	Вертикальное расположение строк

CharSize — размер символов, диапазон изменения составляет от 1 до 10. Стандартное значение для матричного шрифта 8x8 равно единице, для штриховых — четырем. При задании CharSize = 0 шрифт будет выводиться в стандартном размере.

SetTextStyle устанавливает текущий тип шрифта, направление текста и размер символов.

В случае аварийной ситуации процедура устанавливает код ошибки (табл. 8)

Таблица 8

Коды ошибок, устанавливаемые SetTextStyle

GraphResult	Значение
- 8	Не найден файл со шрифтом
- 9	Нет памяти для загрузки шрифта
-11	Ошибка графической системы
-12	Ошибка в/в графической системы
-13	Ошибка в файле со шрифтом
-14	Неверный номер шрифта

Процедура OutText (TextString: String);

где TextString — выводимый текст.

OutText выводит на экран последовательность символов, начиная с текущей позиции. Текущая позиция после выполнения процедуры не изменяется.

Процедура OutTextXY (X, Y: Integer; TextString: String);

где X, Y — исходные координаты;

OutTextXY выводит на экран последовательность символов, начиная с заданных координат (X, Y). Текущая позиция после выполнения процедуры не изменяется.

У процедуры OutTextXY есть следующие особенности. Если строка слишком длинная и выходит за пределы экрана или текущей области просмотра, то она усекается. Если один из активных штриховых шрифтов слишком длинный, то он усекается по границе экрана. Если активен шрифт, используемый по умолчанию (DefaultFont), а строка слишком длинна и не умещается на экране, то она не выводится, поэтому следует точно рассчитывать место для выводимых в графике строк.

Процедура SetTextJustify (Horiz, Vert: Word);

SetTextJustify устанавливает способ выравнивания для вывода текста процедурами OutText и OutTextXY,

где Horiz, Vert — параметры выравнивания по горизонтали и вертикали, которые могут принимать одно из трех объявленных в модуле GRAPH значений:

Для горизонтального ориентирования:

LeftText = 0 — левое выравнивание.
CenterText = 1 — выравнивание по центру.
RightText = 2 — правое выравнивание.

Для вертикального ориентирования:

BottomText = 0 — нижнее выравнивание.
CenterText = 1 — выравнивание по центру.
TopText = 2 — верхнее выравнивание.

Процедура SetUserCharSize (MultX, DivX, MultY, DivY: Word);

где MultX/DivX — коэффициент изменения ширины символов;

MultY/DivY — коэффициент изменения высоты символов.

SetUserCharSize задает коэффициенты изменения ширины и высоты штриховых шрифтов.

Пример с использованием процедур SetTextStyle, SetUserCharSize и OutText:

```
Uses Graph;
  Begin
     INIT;
     SetTextStyle(TriplexFont, HorizDir, 4);
     OutText ('NORM');
     SetUserCharSize (1, 3, 1, 2);
     OutText ('Short');
     SetUserCharSize (3, 1, 1, 1);
     OutText ('Wide');
     ReadIn;
     CloseGraph;
  End.
```

До того, как осуществлять вывод текстовой строки, полезно знать размеры области, которую займет эта строка на экране. Эту информацию можно получить с помощью следующих функций:

Функция TextHeight (TextString: String): Word;

TextHeight возвращает высоту строки в пикселях.

Функция TextWidth (TextString: String): Word;

TextWidth возвращает ширину строки в пикселях.

Для получения характеристик текста используют процедуру:

Процедура GetTextSettings (Var SetInfo: TextSettingsType);

Для этого определен следующий тип:

```
Тип TextSettingsType = record
Font: Word; {номер шрифта}
Direction: Word; {направление}
CharSize: Word, {размер шрифта}
Horiz: Word, {ориентация по x}
Vert: Word; {ориентация по y}
End;
```

Пример:

```
Uses Graph;
  Var OldStyle: TextSettingsType,
  Begin
     INIT; {процедура инициализации}
     {Заносим в перем. OldStyle текущие установки
     для шрифта}
     GetTextSettings(OldStyle);
     OutTextXY (0, 0,' Old text style');
     SetTextJustify(LeftText, CenterText); {выравнивание
     текста}
     {Установка новых параметров для шрифта}
     SetTextStyle (TriplexFont, VertDir, 4);
     OutTextXY (GetMaxX div 2, GetMaxY div 2,' New Style');
     With OldStyle do
Begin
     SetTextJustify (Horiz, Vert);
     {Установка старых параметров для шрифта}
     SetTextStyle(Font, Direction, CharSize);
End;
OutTextXY (0, TextHeight ('H'), ' Old style again');
ReadIn;
CloseGraph;
End.
```

ГРАФИКА В СИСТЕМЕ ТУРБО ПАСКАЛЬ

Составитель *Логвинова Елена Александровна*

Редактор Т. К. Кретинина
Техн. редактор Н. М. Каленюк
Корректор Т. И. Щелокова
Компьютерная верстка О. А. Карасева

Подписано в печать 13.02.95 . Формат 60x84 1/16.
Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,16. Усл. кр.-отт. 1,28. Уч.-изд. л. 1,2.
Тираж 200 экз. Заказ 117. Арт. С—74/95

Самарский государственный аэрокосмический
университет имени академика С. П. Королева.
443086 Самара, Московское шоссе, 34.

Издательство Самарского аэрокосмического университета.
443001, г. Самара, ул. Ульяновская, 18