Министерство науки, высшей школы
и технической политики Российской Федерации

Самарский государственный аэрокосмический
университет имени академика С. П. Королева

ГРАФИКА
В СИСТЕМЕ ТУРБО-СИ

Методические указания
к лабораторным работам

САМАРА 1992

Составители: А.А.Макаров, В.В.Семенов

УДК 681.3.06

Графика в системе Турбо-Си: Метод.указания к лабораторным работам, Самар.аэрокосм.ун-т; Сост. А.А.Макаров, В.В.Семенов. Самара,1992. 18 с.

Рассматриваются следующие вопросы: состав графической системы в Турбо-Си, инициализация графического режима, наиболее часто используемые графические функции.

Предназначены для проведения занятий со слушателями ФПК ИТР, а также для студентов, использующих машины типа IBM PC.

Составлены на кафедре "Программное обеспечение вычислительных систем".

Печатаются по решению редакционно-издательского совета Самарского Государственного аэрокосмического университета имени академика С.П.Королева

Рецензент А.В.Баландин

НАЧАЛЬНЫЕ СВЕДЕНИЯ

Графика в системе Турбо-Си поддерживается следующими файлами:

graphics.h – файл определений для графического режима;

graphics.lib – библиотека графических функций;

att.bgi, cga.bgi, egavga.bgi, herc.bgi, ibm8514.bgi, pc3270.bgi – графические драйверы;

goth.chr, litt.chr, sans.chr, trip.chr -драйверы шрифтов в графическом режиме.

Перед началом работы программы, использующей графические функции, должна быть выполнена инициализация графической системы, т.е. указан графический драйвер и режим его работы. Имя драйвера и номер режима задаются в виде констант,определенных в файле graphics.h. Выбор того или иного драйвера определяется типом установленного в компьютере видео-адаптера(CGA,EGA,VGA и др.). От этого зависит и графический режим работы - размер экрана в пикселах, количество цветов, графических страниц. Обычно используется режим автоматического определения типа видеоадаптера и , соответственно, осуществляется выбор нужного графического драйвера. Это можно сделать, например, с помощью следующей функции:

```
#include <graphics.h>
int    GraphDriver;  /* Графический драйвер */
int    GraphMode;/* Режим работы графического драйвера */
int    ErrorCode;                /* Код ошибки инициализации */
void Initialize(void)
{ GraphDriver = DETECT; /* Автоматическое определение типа драйвера */
  initgraph( &GraphDriver, &GraphMode, "" ); /* Третий параметр -
                        строка,указывающая маршрут к драйверу */
   ErrorCode=graphresult( );/*Считывание результата инициализации */
  if( ErrorCode != grOk ){
   printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ));
    exit(1);}}
```

Приведем основные графические функции Турбо-Си. За начало координат(x=0,y=0) принимается верхний левый угол экрана. Значение x определяет число точек, на которое смещается вправо указатель текущей позиции экрана от начала координат. Значение y определяет аналогичное смещение вниз.

При установке на экране графического окна за начало координат принимается верхний левый угол этого окна, который может быть смещен относительно верхнего левого угла экрана. Угловые величины задаются в градусах. Все функции разбиты по их функциональному назначению.-

ИНИЦИАЛИЗАЦИЯ И ЗАКРЫТИЕ ГРАФИЧЕСКОЙ СИСТЕМЫ

```
void closegraph(void);
```
закрывает графическую систему и осуществляет переход в текстовый режим.

```
void detectgraph(int far *graphdriver, int far *graphmode);
```
определяет тип графического драйвера и режим его работы:
graphdriver - тип графического драйвера,
graphmode - режим работы драйвера.

```
void initgraph(int far *graphdriver,int far *graphmode,
     char far *pathtodriver);
```
инициализирует графическую систему:
pathtodriver - маршрут к графическому драйверу.

```
int installuserdriver(cha. far *name, int huge (*detect) (void));
```
устанавливает пользовательский драйвер формата BGI.

```
int installuserfont(char far *name);
```
загружает файл пользовательского шрифта.

```
char *getdrivername(void);
```

возвращает указатель на строку, идентифицирующую текущий драйвер.

int getmaxmode(void);

возвращает максимальный номер графического режима для данного драйвера.

char *getmodename(int mode_number);

возвращает указатель на строку, идентифицирующую графический режим mode_number.

void getmoderange(int grdriver, int far *lomode, int far *himode);

для данного драйвера возвращает соответственно наименьший и наибольший номер графического режима.

int registerbgidriver(void(*driver)(void));

регистрирует в графической системе загруженный пользователем или скомпанованный с программой драйвер формата BGI(формат фирмы Borland). При наличии ошибок, возвращаемое значение меньше 0. В противном случае, возвращается внутренний номер драйвера.

int registerbgifont(void(*font)(void));

регистрирует загружаемый пользователем или скомпанованный с программой шрифт формата BGI(формат фирмы Borland) в графической системе.

В случае ошибки, возвращает значение меньше 0. В противном случае, возвращает внутренний номер шрифта.

void restorecrtmode(void);

возвращает режим экрана в исходное состояние(то, которое существовало до инициализации графического режима).

unsigned setgraphbufsize(unsigned bufsize);

изменяет размер внутреннего графического буфера:
bufsize - новый размер(в байтах). Функция возвращает предыдущий размер внутреннего графического буфера.

void setgraphmode(int mode);

очищает экран, устанавливает графическую систему в режим, заданный параметром mode. Например, для адаптера VGA:
VGALO=0 - (640x200 пикселов,16 цветов,4 графические страницы),
VGAMED=1 - (640x350 пикселов,16 цветов,2 графические страницы),
VGAHI=2 - (640x480 пикселов,16 цветов,1 графическая страница).
Все возможные режимы для различных адаптеров перечислены в файле graphics.h.

УСТАНОВКА ПАРАМЕТРОВ ИЗОБРАЖЕНИЯ

void graphdefaults(void);

все графические установки возвращает в исходное состояние.

void setactivepage(int page);

устанавливает для графического вывода активную страницу.Страница, заданная параметром page становится активной. Весь графический вывод будет направляться в нее.

void setallpalette(struct palettetype far *palette);

изменяет все цвета палитры:

```
struct palettetype {
      unsigned char size;
      signed char colors[MAXCOLORS+1];};
```

void setaspectratio(int xasp, int yasp);

устанавливает коэффициент приведения линий по x и y к одинаковой длине.

void setbkcolor(int color);

устанавливает цвет фона. Значение color может быть установлено в диапазоне от 0 до 15 (VGA).

```
void setcolor(int color);
```

устанавливает цвет проводимых линий.

```
void setfillstyle(int pattern,int color);
```

устанавливает цвет color для наполнителя вида pattern.
Используется для задания стиля наполнителя при построении закрашенных фигур. Возможные значения параметра pattern:

EMPTY_FILL	=0,	HATCH_FILL	=7,
SOLID_FILL	=1,	XHATCH_FILL	=8,
LINE_FILL	=2,	INTERLEAVE_FILL	=9,
LTSLASH_FILL	=3,	WIDE_DOT_FILL	=10,
SLASH_FILL	=4,	CLOSE_DOT_FILL	=11,
BKSLASH_FILL	=5,	USER_FILL	=12.
LTBKSLASH_FILL	=6,		

```
void setfillpattern(char far *upattern,int color);
```

устанавливает пользовательский заполнитель экрана,где upattern - указатель на массив, в котором определена конфигурация элемента заполнения 8x8 точек(8 байт).

```
void setlinestyle(int linestyle,unsigned upattern,int thickness);
```

устанавливает стиль линии:
linestyle - SOLID_LINE=0,
DOTTED_LINE=1,
CENTER_LINE=2,
DASHED_LINE=3,
USERBIT_LINE=4;
толщина линии:
thickness - NORM_WIDTH=1 нормальная,
THICK_WIDTH=3 утолщенная,
способ получения:
для стандартных стилей (linestyle от 0 до 3) upattern=0,
для linestyle=USERBIT_LINE значение параметра upattern влияет на устанавливаемый пользователем стиль.

```
void setpalette(int colornum, int color);
```
изменяет один цвет из палитры.

```
void setrgbcolor(int colornum,int red,int green, int blue);
```
устанавливает палитру цветов для адаптеров VGA и IBM-8514.

```
void setrgbpalette(int colornum,int red,int green, int blue);
```
устанавливает цвета(256) для адаптера IBM-8514.

```
void settextjustify(int horiz,int vert);
```
устанавливает способ выравнивания текста:

horiz - LEFT_TEXT =0 выровнять влево,
CENTER_TEXT=1 центрировать,
RIGHT_TEXT =2 выровнять вправо;

vert - BOTTOM_TEXT=0 переместить вниз,
CENTER_TEXT=1 центрировать,
TOP_TEXT =2 переместить вверх.

```
void settextstyle (int font,int direction,int charsize);
```
устанавливает стиль текста:

font (шрифт) - DEFAULT_FONT =0,
TRIPLEX_FONT =1,
SMALL_FONT =2,
SANS_SERIF_FONT=3,
GOTHIC_FONT =4;

direction(направление) - HORIZ_DIR=0 слева направо,
VERT_DIR =1 снизу вверх;

charsize - размер символа(цифра, указывающая, во сколько раз увеличивается его величина. Если она нуль, то размер задается пользователем с помощью функции setusercharsize.)

```
void setusercharsize(int multx,int divx,int multy,int divy);
```
устанавливает размер символа пользователя - увеличение по горизонтали в multx/divx раз. по вертикали в multy/divy раз.

void setviewport(int left,int top,int right,int bottom,int clip);

устанавливает размер окна для вывода графического изображения:
left,top - координаты верхней левой точки,
right,bottom - координаты нижней правой точки.
Если clip=1, то происходит отсечение изображения за пределами окна , если clip=0, то отсечения не происходит.

void setvisualpage(int page);

задает номер отображаемой графической страницы. Страница, определяемая параметром page становится отображаемой графической страницей.

void setwritemode(int mode);

устанавливает режим вычерчивания линий:
mode=0 - линия пересекает существующую картинку,
mode=1 - линия не проходит по местам, "занятым" существующими пикселами.

ПОСТРОЕНИЕ ИЗОБРАЖЕНИЙ

void arc(int x,int y,int stangle,int endangle,int radius);

вычерчивает дугу окружности:
x,y - координаты центра окружности,
radius - ее радиус,
stangle,endangle - начальный и конечный угол(отсчет против часовой стрелки).

void bar(int left,int top,int right,int bottom);

вычерчивает закрашенный прямоугольник(цвет и образец закраски устанавливаются функцией setfillstyle()).

void bar3d(int left,int top,int right,int bottom,int depth,int topflag);

вычерчивает прямоугольный параллелепипед с закрашенной лицевой гранью.
Если topflag=0, то верхняя грань отсутствует, если topflag=1-нет, depth - глубина.

```
void circle(int x,int y, int radius);
```
вычерчивает окружность радиуса radius в точке с координатами x,y.

```
void cleardevice(void);
```
очищает графический экран.

```
void clearviewport(void);
```
очищает ранее установленное окно графического экрана.

```
void drawpoly(int numpoints, int far polypoints[]);
```
вычерчивает многоугольник :
polypoints - массив структур, в которых определены координаты x и y всех узловых точек,
numpoints - количество этих точек.

```
void ellipse(int x,int y,int stangle,int endangle,int xradius,
        int yradius);
```
вычерчивает эллипс.

```
void fillellipse(int x,int y,int xradius,int yradius);
```
вычерчивает и заполняет выбранным образцом эллиптическую дугу.

```
void fillpoly(int numpoints, int far polypoints[]);
```
вычерчивает и закрашивает многоугольник (параметры определяются также как и в drawpoly).

```
void floodfill(int x,int y,int border);
```
закрашивает установленным ранее наполнителем область экрана, в которую попадает точка с координатами x,y(если border=1, заполнение выполняется,если border=0 - нет).

void line(int x0,int y0,int x1,int y1);
вычерчивает линию из точки с координатами x0,y0 в точку с координатами x1,y1.

void linerel(int dx,int dy);
вычерчивает линию из текущей позиции(CP) в точку с относительными координатами x,y.

void lineto(int x,int y);
проводит линию из CP в точку с абсолютными координатами x,y.

void moverel(int dx,int dy);
перемещает указатель CP из заданной точки в точку с относительными координатами x,y.

void moveto(x,y);
перемещает указатель CP в точку x,y.

void outtext(char far *textstring);
выводит текстовую строку textstring в текущую позицию графического экрана.

void outtextxy(x,y,char far *textstring);
выводит текстовую строку textstring с координатами x,y(относительно них осуществляется выравнивание).

void pieslice(int x,int y, int stangle,int endangle, int radius);
вычерчивает сектор круга с заполнением.

void putimage(int x,int y, void far *bitmap,int op);
выводит сохраненное графическое изображение в окно экрана:
bitmap - указатель на область памяти, где хранится графическое изображение(см.getimage()),
op - COPY_PUT =0.

```
XOR_PUT   =1,
OR_PUT    =2,
AND_PUT   =3,
NOT_PUT   =4  константа, определяющая способ наложения
```

выводимого окна на другое изображение экрана,
x,y - координата начальной точки выводимой картинки.

```
void putpixel(int x,int y,int pixelcolor);
```

выводит точку цвета pixelcolor в координату x,y.

```
void rectangle(int left,int top, int right, int bottom);
```

вычерчивает прямоугольник.

```
void sector(int x,int y, int stangle,int endangle, int xradius,
     int yradius);
```

вычерчивает эллиптический сектор с заполнением.

ПОЛУЧЕНИЕ ПАРАМЕТРОВ ИЗОБРАЖЕНИЯ

```
void getarccoords(struct arccoordstype far *arccoords);
```

дает значения координат дуги, являющихся элементами структуры типа arccoordstype,которые получены в результате последнего вызова функции arc.

```
struct arccoordtype  { /* параметры дуги */
       int x,y;
       int xstart,ystart,xend,yend;};/* координаты начала и конца
                                        дуги */

void getaspectratio(int far *xasp, int far *yasp);
```

получает коэффициент(yasp/xasp) приведения линий по координатам x и y к одинаковой длине.

```
int getbkcolor(void);
```

возвращает номер текущего цвета фона.

int getcolor(void);

возвращает номер текущего цвета линий.

struct palettetype *far getdefaultpalette(void);

возвращает структуру, описывающую палитру для текущего драйвера.

void getfillpattern(char far *pattern);

получает текущий образец заполнителя, заданного последним обращением к функции setfillpattern( ).

void getfillsettings(struct fillsettingstype far *fillinfo);

дает значения параметров заполнения и цвета экрана,являющихся элементами структуры типа fillsettingstype:

```
struct fillsettingstype { /* параметры заполнения */
  int pattern;            /* наполнитель */
  int color; };           /* цвет */
```

int getgraphmode(void);

возвращает текущую графическую моду для данного видеоадаптера.

void getimage(int left,int top,int right,int bottom, void far *bitmap);

получает и сохраняет в области памяти, на которую указывает bitmap окно экрана с заданным размером.

void getlinesettings(struct linesettingstype far *lineinfo);

дает значения параметров линии, являющихся элементами структуры типа linesettingstype:

```
struct linesettingstype {   /* параметры линии */
   int linestyle;           /* стиль линии */
   unsigned upattern;       /* способ получения */
   int thickness; };        /* толщина */
```

```
int  getmaxx(void);
```
возвращает целое значение, равное размеру экрана по горизонтали (в пикселах).

```
int  getmaxy(void);
```
возвращает целое значение, равное размеру экрана по вертикали(в пикселах).

```
void getpalette(struct palettetype far *palette);
```
возвращает в структуре типа palettetype информацию о текущей палитре(см. функцию setallpalette()).

```
int getpalettesize(void);
```
возвращает число цветов в палитре для данных драйвера и режима.

```
int  getpixel(int x,int y);
```
возвращает цвет точки с координатами x,y.

```
void gettextsettings(struct textsettingstype far *textinfo);
```
дает значения параметров текста, сохраняемых в структуре, на которую указывает textinfo:

```
struct textsettingstype {   /* параметры текста */
   int font;                /* шрифт */
   int direction;           /* направление */
   int charsize;            /* размер символа */
   int horiz;               /* выравнивание по горизонтали */
   int vert;};              /* выравнивание по вертикали */
```

```
void getviewsettings(struct viewporttype *vp);
```
устанавливает указатель vp на структуру, в которой хранятся параметры окна графического экрана:

```
struct viewporttype {        /* параметры окна экрана */
   int left,top,right,bottom;/* слева,сверху,справа,снизу */
   int clip;};               /* отсечение */
```

int getx(void);

возвращает координату x для CP.

int gety(void);

возвращает координату y для CP.

char *grapherrormsg(int errorcode);

возвращает указатель на строку, ассоциированную с номером ошибки, возвращаемым функцией graphresult().

int graphresult(void);

возвращает номер ошибки(целое число от 1 до 15). Значение 0 говорит об отсутствии ошибок.

unsigned imagesize(int left,int top,int right,int bottom);

дает значение объема буфера(в байтах), который необходим для сохранения графической информации в окне заданного размера.

int textheight(char far *textstring);

дает целое значение высоты символа, переданного в виде параметра.

int textwidth(char far *textstring);

дает целое значение ширины символа, передаваемого в виде параметра.

В заключение покажем простейшую программу на Си, рисующую окружность заданного радиуса.

```
#include <stdio.h>
#include <graphics.h>
  main()
 { int r,gd=DETECT,gm;
    printf(" введите радиус окружности: ");
     scanf("%d",&r);
      initgraph(&gd,&gm,"");
       circle(getmaxx()/2,getmaxy()/2,r);
```

```
    getch( );
      closegraph( ); }
```

Примечание: при работе с Turbo-C (v.2), Turbo-C++, Borland-C++ для включения графической библиотеки в состав выполняемого модуля следует в главном меню установить соответствующий режим работы компановщика.

ГРАФИКА В
СИСТЕМЕ ТУРБО-СИ

Составители: М а к а р о в Алексей Алексеевич
С е м е н о в Валерий Владимирович

Редактор Е.Д.Антонова
Техн.редактор Г.А.Усачева

Подписано в печать 12.II.92. Формат 60x84 1/16.
Бумага офсетная. Печать офсетная. Усл.печ.л. 0,93.
Усл.-изд.л. 0,8. Тираж 100 экз.
Заказ N 216 Арт.С-18/92.

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

443001, г.Самара ИПО. Участок оперативной полиграфии
Самарского Государственного аэрокосмического университета,
ул. Ульяновская, 18.