

Лекция 7

ГРАФИЧЕСКИЕ

СРЕДСТВА

Возможности

Стандартная графическая библиотека
(Приложения **Standard Graphics** и **Quick Win**)

Графическая подсистема **Windows
Graphics Device Interface**
(Приложения **Console, Windowing Application**)

+

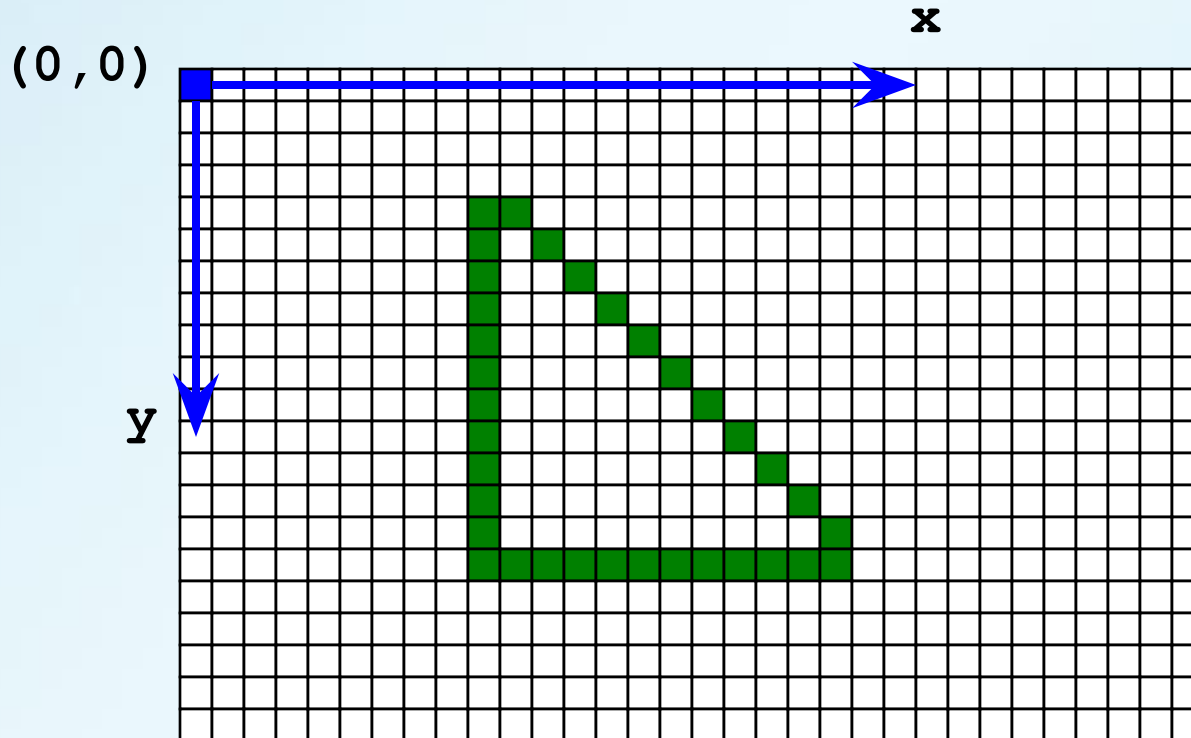
Графические библиотеки (**OpenGL, SciGraph**)

Стандартная графика

Модуль `IFQWIN`

Типы данных, константы, интерфейсы процедур.

После оператора `program` указать `use IFQWIN`



Конфигурация окна

Логическая функция `flag = SetWindowConfig(wc)`
устанавливает конфигурацию окна

```
type (WINDOWCONFIG)
  INTEGER*2  NUMXPixels      ! число пикселей по оси X
  INTEGER*2  NUMYPixels      ! число пикселей по оси Y
  INTEGER*2  NUMTEXTCOLS    ! число столбцов текста
  INTEGER*2  NUMTEXTROWS    ! число строк текста
  INTEGER*2  NUMCOLORS      ! количество цветов
  INTEGER*4  FONTSIZE        ! размер шрифта
  CHARACTER*(80) TITLE      ! заголовок окна (Си-строка)
  INTEGER*2  BITSPERPIXEL   ! бит на пиксел
  . . .
end type
```

Логическая функция `flag = GetWindowConfig(wc)`
получает конфигурацию окна

Конфигурация окна

Получить текущее разрешение окна приложения,
установить заголовок окна.

```
program graph1
use ifqwin

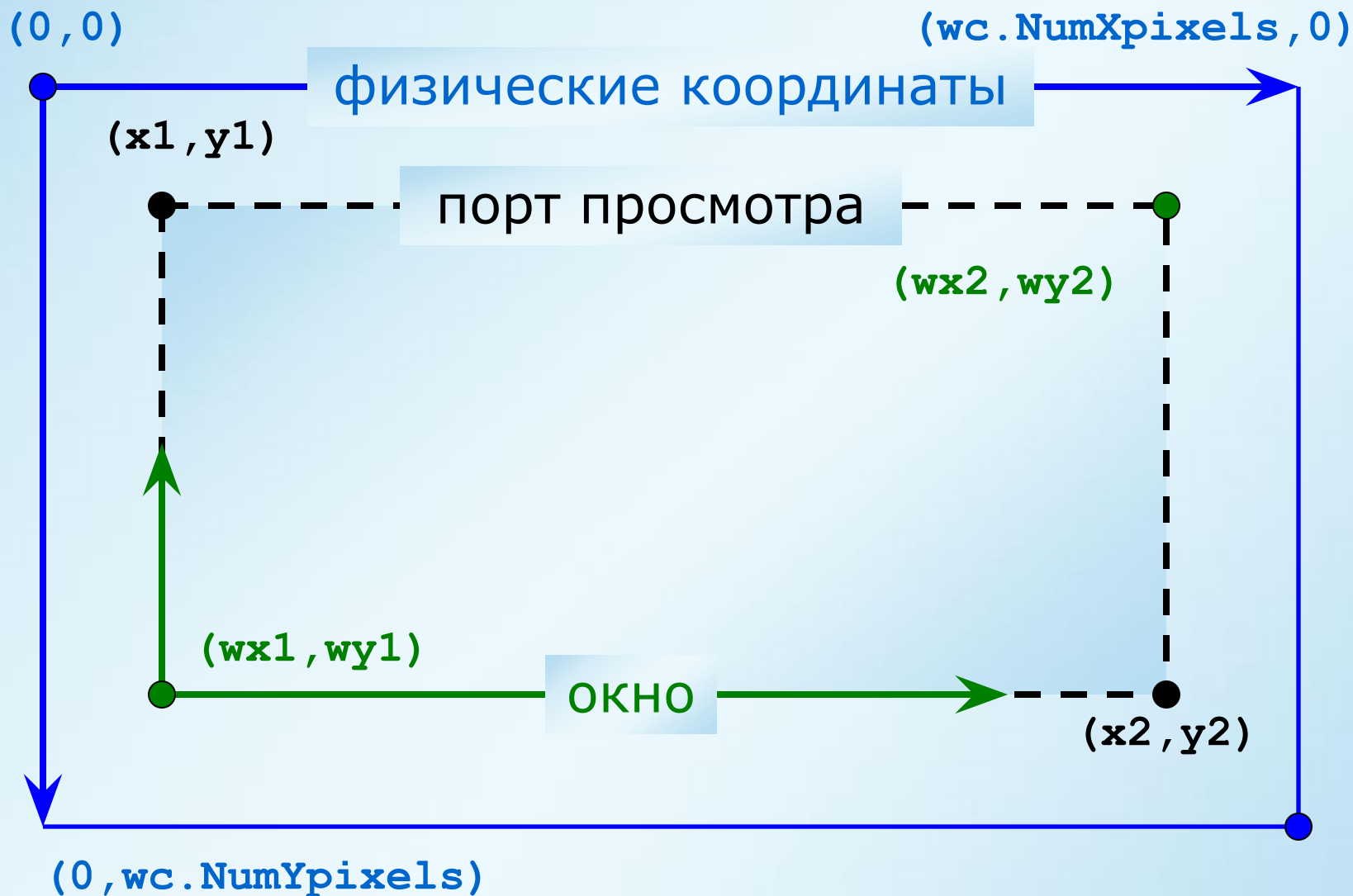
logical(4) flag
type (windowconfig) wc

flag = GetWindowConfig(wc)
write(*,*) "Number of X pixels = ", wc.NUMXPIXELS
write(*,*) "Number of Y pixels = ", wc.NUMYPIXELS

wc.title = "Первая программа"
flag = SetWindowConfig(wc)

end
```

Системы координат



Системы координат

`call SetViewOrg(xp, yp, t)` – перенос начала координат $(0, 0)$ в точку физической системы координат (xp, yp) .

`call SetClipRgn(x1, y1, x2, y2)` – ограничение области вывода данных.

`call SetViewPort(x1, y1, x2, y2)` – установка порта просмотра.

`res = SetWindow(finvert, wx1, wy1, wx2, wy2)` – установка окна.

Системы координат

Пример. Создать окно для построения графика функции $y(x) = \sin(x)$, на отрезке от 0.0 до 3.0.


```
program graph2
use ifqwin
logical(4) flag
integer(4) ires4
type (windowconfig) wc

flag = GetWindowConfig(wc)
wc.title = "Оконная система координат"
flag = SetWindowConfig(wc)

call SetViewport(100,100,&
INT2(wc.NumXpixels-100),INT2(wc.NumYpixels-100))

ires4 = SetWindow(.TRUE.,0.0d0,0.0d0,0.0d0,3.0d0)
end
```


Стандартная палитра 16 цветов



| | | | |
|----|---|-------------------------------|--------------------|
| 0 | – | <code>\$BLACK</code> , | черный; |
| 1 | – | <code>\$BLUE</code> , | синий; |
| 2 | – | <code>\$GREEN</code> , | зеленый; |
| 3 | – | <code>\$CYAN</code> , | голубой; |
| 4 | – | <code>\$RED</code> , | красный; |
| 5 | – | <code>\$MAGENTA</code> , | фиолетовый; |
| 6 | – | <code>\$BROWN</code> , | коричневый; |
| 7 | – | <code>\$WHITE</code> , | белый; |
| 8 | – | <code>\$GRAY</code> , | серый; |
| 9 | – | <code>\$LIGHTBLUE</code> , | светло-синий; |
| 10 | – | <code>\$LIGHTGREEN</code> , | светло-зеленый; |
| 11 | – | <code>\$LIGHTCYAN</code> , | светло-голубой; |
| 12 | – | <code>\$LIGHTRED</code> , | светло-красный; |
| 13 | – | <code>\$LIGHTMAGENTA</code> , | светло-фиолетовый; |
| 14 | – | <code>\$YELLOW</code> , | желтый; |
| 15 | – | <code>\$LIGHTWHITE</code> , | ярко-белый. |

Управление цветом

Стандартная 16-цветная палитра.

`ires2 = SetBkColor(color)` - цвет фона.

`ires2 = SetColor(color)` - цвет рисования.

Произвольный **RGB**-цвет.

`ires4 = SetBkColorRGB(color)` - цвет фона.

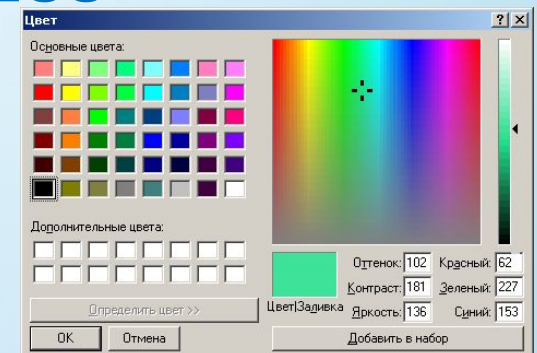
`ires4 = SetColorRGB(color)` - цвет рисования.

`color = RGBToInteger(R,G,B)`

$R = 0..255, G = 0..255, B = 0..255$

Очистка экрана

`call ClearScreen(area)`



Управление стилем линий

```
integer(2) mask_solid(16), &  
          mask_dash(16)
```

! --- сплошная

```
mask_solid = (/2#1111111111111111/)
```

! --- штрихи

```
mask_dash  = (/2#1111000011110000/)
```

Изменение типа линий

```
call SetLineStyle(mask)
```

Управление маской заполнения

! --- маска заполнения "точки"

```
integer(1) maska(8)
maska=(/2#00000000, &
      2#00000000, &
      2#00111100, &
      2#00111100, &
      2#00111100, &
      2#00111100, &
      2#00000000, &
      2#00000000/)
```

Изменение маски заполнения

```
call SetFillMask(maska)
```

Графические примитивы

- 1) Пиксел,
- 2) отрезок прямой линий,
- 3) прямоугольник,
- 4) многоугольник,
- 5) эллипс (окружность),
- 6) дуга окружности,
- 7) сектор,
- 8) произвольная замкнутая область.

Все графические примитивы при рисовании отображаются текущим цветом рисования, типом линии и маской заполнения.

Графические примитивы

Отдельный пиксел
(физическая система координат)

```
ires2 = SetPixel(x,y)
```

```
ires4 = SetPixelRGB(x,y,color)
```

(оконная система координат)

```
ires2 = SetPixel_W(x,y)
```

```
ires4 = SetPixelRGB_W(x,y,color)
```

Группа пикселов

```
call SetPixels(N,X,Y,COLOR)
```

N – число элементов

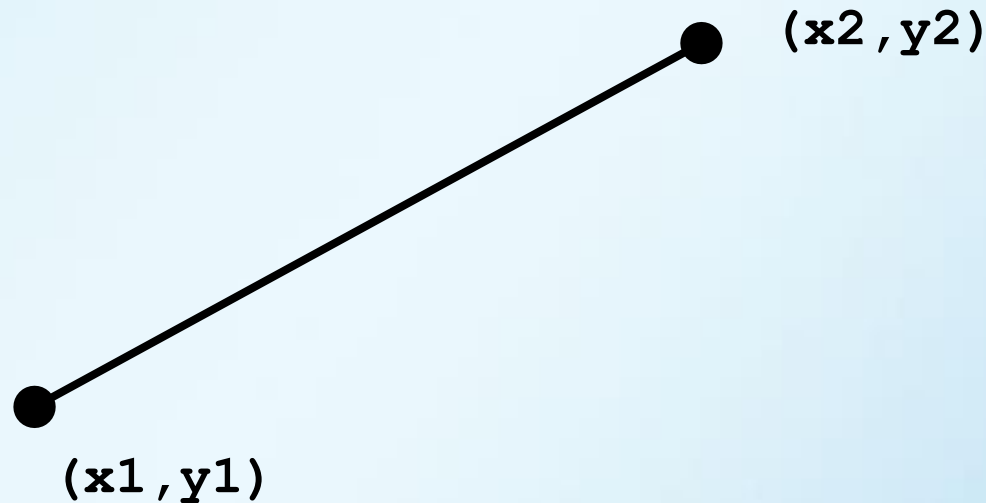
X,Y – массивы точек

COLOR – массив цветов

Графические примитивы

Отрезок прямой линии

```
call MoveTo (x1, y1, xy)  
ires2 = LineTo (x2, y2, color)
```



Графические примитивы

Прямоугольник

```
ires2 = Rectangle(control, x1, y1, x2, y2)
```

control:

\$GFILLINTERIOR - заливка;

\$GBORDER - границы.



Графические примитивы

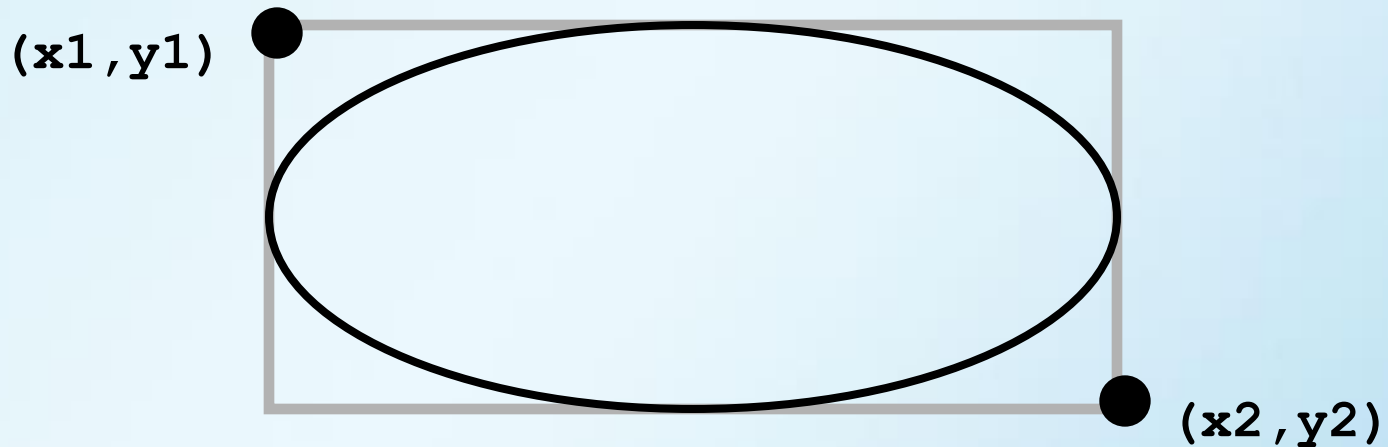
Эллипс

```
ires2 = Ellipse(control, x1, y1, x2, y2)
```

control:

\$GFILLINTERIOR - заливка;

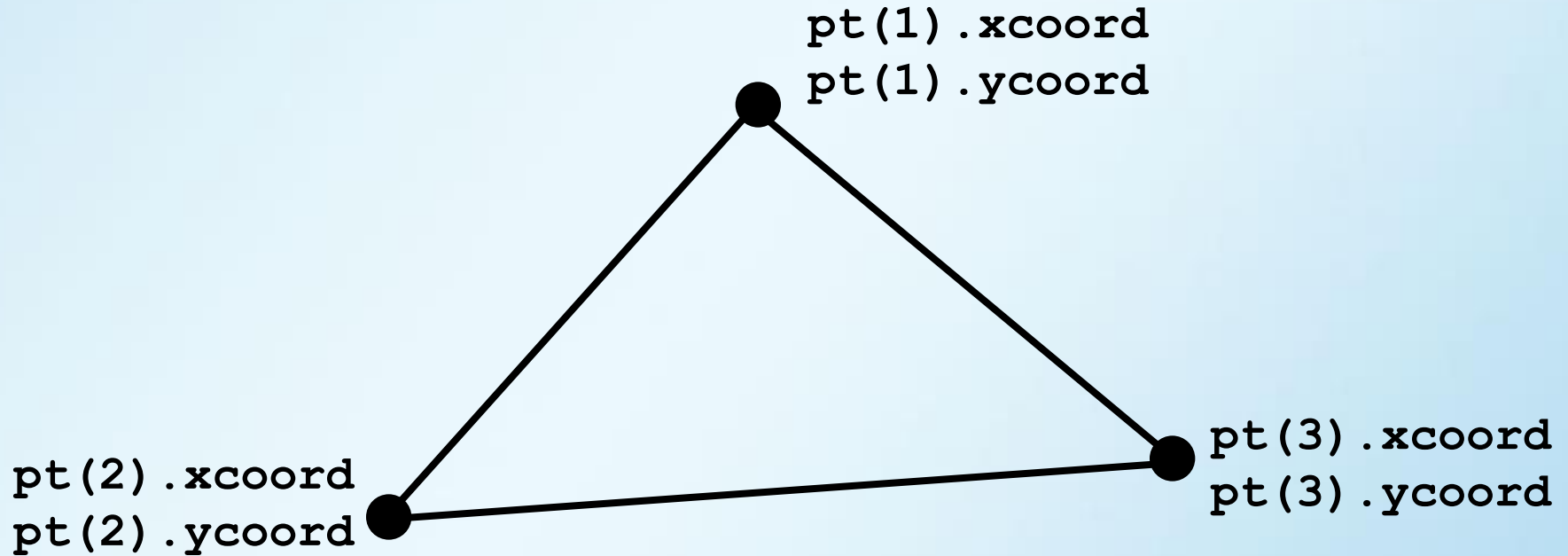
\$GBORDER - границы.



Графические примитивы

Многоугольник

```
ires2 = Polygon(control,pt,N) , где  
integer(2)    ires2, control  
type (xcoord) pt ! массив вершин  
integer(2)    N   ! число вершин
```

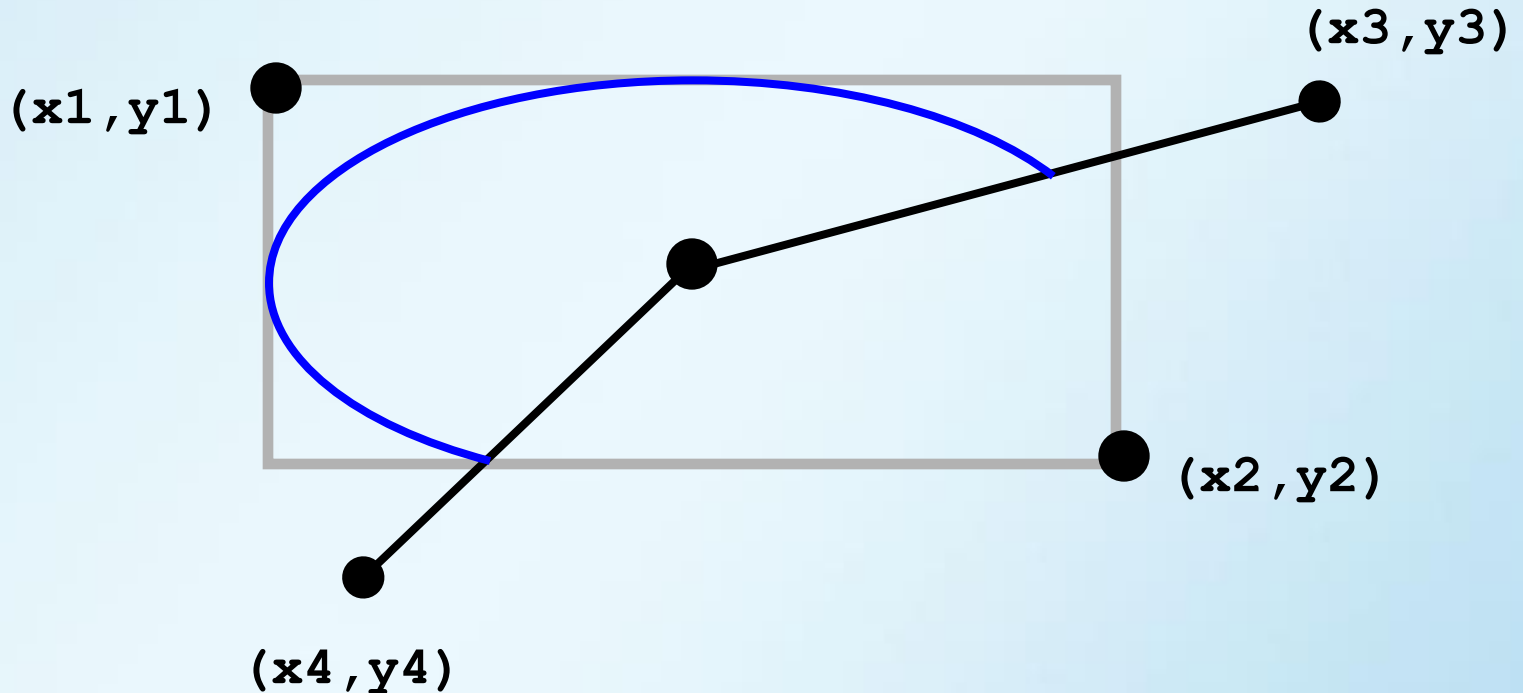


Графические примитивы

Дуга и сектор эллипса

```
ires2 = Arc (x1, y1, x2, y2, x3, y3, x4, y4)
```

```
ires2 = Pie (control, x1, y1, x2, y2, x3, y3, x4, y4)
```

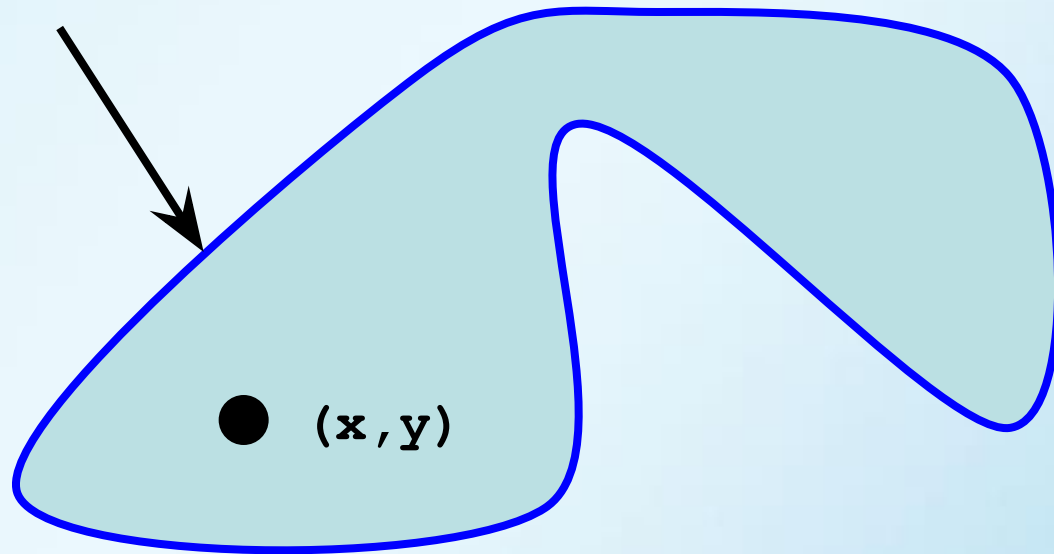


Графические примитивы

Произвольная замкнутая область

```
ires2 = FloodFill(x, y, bordercolor)
```

`bordercolor`
(цвет сплошной границы)



Обработка изображений

Запись изображения в **bmp**-файл

```
ires4 = SaveImage (FileName, x1, y1, x2, y2)
```

Чтение изображения из **bmp**-файла

```
ires4 = LoadImage (FileName, x, y)
```

Пример Standard Graphics (1)

```
program picture ! Графический сборник
use ifqwin
type(xcoord) pt(3)
integer(2) ires2
integer(1) :: mask(8)=[B'00000000', &
                       B'00000000', &
                       B'00111100', &
                       B'00111100', &
                       B'00111100', &
                       B'00111100', &
                       B'00000000', &
                       B'00000000']

ires2 = setcolor(9)
ires2 = ellipse($GBORDER,70,70,230,230) ! окружность
pt(1).xcoord = 450; pt(1).ycoord = 10 ! треугольник
pt(2).xcoord = 320; pt(2).ycoord = 120
pt(3).xcoord = 600; pt(3).ycoord = 300
```

Пример Standard Graphics (2)

```
ires2 = Polygon($GBORDER,pt,3)
```

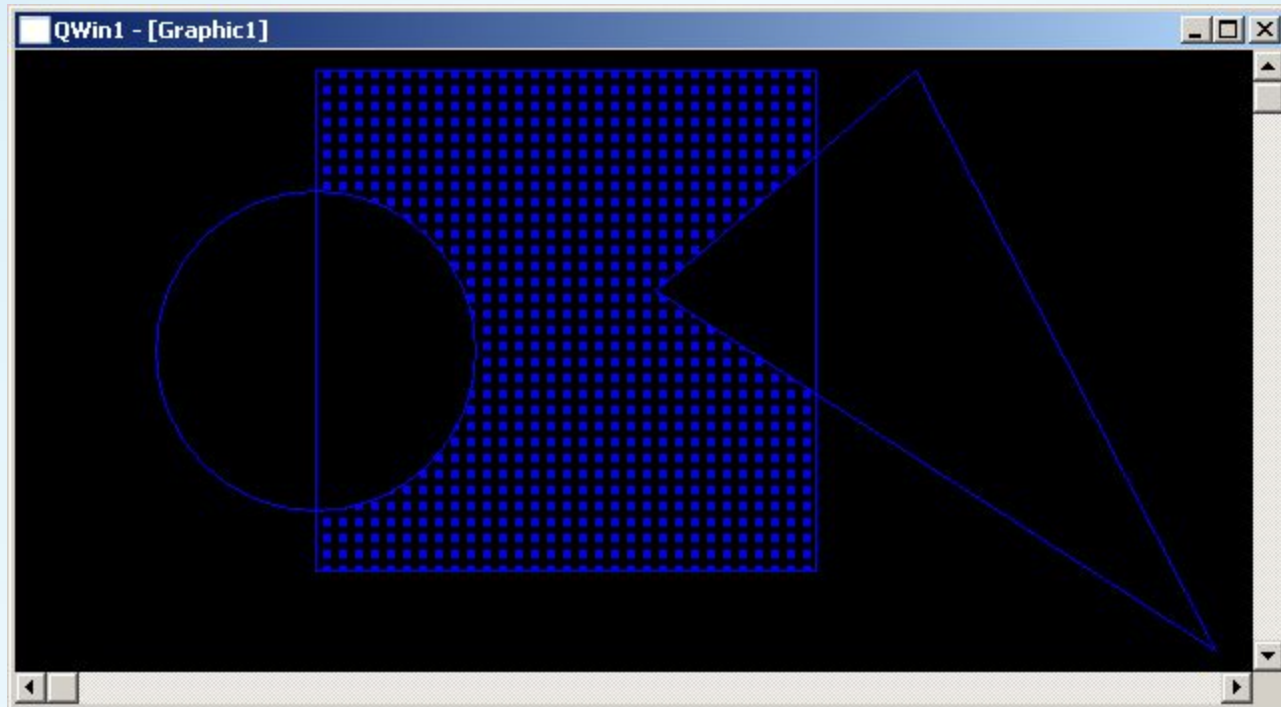
```
ires2 = Rectangle($GBORDER,150,10,400,260) ! прямоугольник
```

```
call SetFillMask(mask)
```

```
ires2 = FloodFill(320,240,9) ! замкнутая область
```

```
ires4 = SaveImage("D:\1.bmp",1,1,400,300) ! сохранение
```

```
end
```



Используем GDI

GDI (Graphics Device Interface)
подсистема **Windows**,
отвечающая за вывод графики и текста.

Работа с **GDI** аналогична работе
с процедурами **Standard Graphics**.

GDI предоставляет в разы больше возможностей.

Контекст устройства

Контекст устройства

DC (Device Context) – структура данных, содержащая параметры и атрибуты вывода графики на устройство.

5 типов контекста устройства:

- дисплей (**Display DC**);
- принтер (**Printer DC**);
- память (**Memory DC**);
- метафайл (**Metafile DC**);
- информационный (**Information DC**).

Графические объекты

- перо (**pen**)
вывод линий (цвет, толщина, стиль);
- кисть (**brush**)
закраски фигур (цвет, стиль);
- шрифт (**font**)
свойства шрифта, для вывода текста;
- палитра (**palette**)
набор используемых в **DC** цветов;
- область (**region**)
задает области отсечения (**clipping regions**),
вне которых вывод графики блокируется.

Работа с дескрипторами

Работа с графическими объектами при помощи дескрипторов (**handles**).

HDC, дескриптор контекста

HPEN, дескриптор пера

HBRUSH, дескриптор кисти

HFONT дескриптор шрифта

...

Создание и удаление объектов производится с помощью соответствующих функций.

Процедуры создания

Перо (карандаш)

```
hPen = CreatePen(STYLE, width, RGB(R,G,B));
```

STYLE : PS_SOLID – сплошная линия

PS_DASH – штрихи

PS_DOT – пунктир

PS_DASHDOT – штрих пунктир

width – толщина, 0 – один пиксел

R, G, B – интенсивность цвета 0..255

Кисть (заливка)

```
hBrush = CreateSolidBrush(RGB(R, G, B));
```

Процедуры рисования

Пиксел

```
ires = SetPixel(hdc, x, y, RGB(R, G, B))
```

Отрезок

```
ires = MoveToEx(hdc, x1, y1, NULL)  
ires = LineTo(hdc, x2, y2)
```

Прямоугольник

```
ires = Rectangle(hdc, x1, y1, x2, y2)
```

Эллипс

```
ires = Ellipse(hdc, x1, y1, x2, y2)
```

Дуга эллипса

```
ires = Arc(hdc, x1, y1, x2, y2, x3, y3, x4, y4)
```

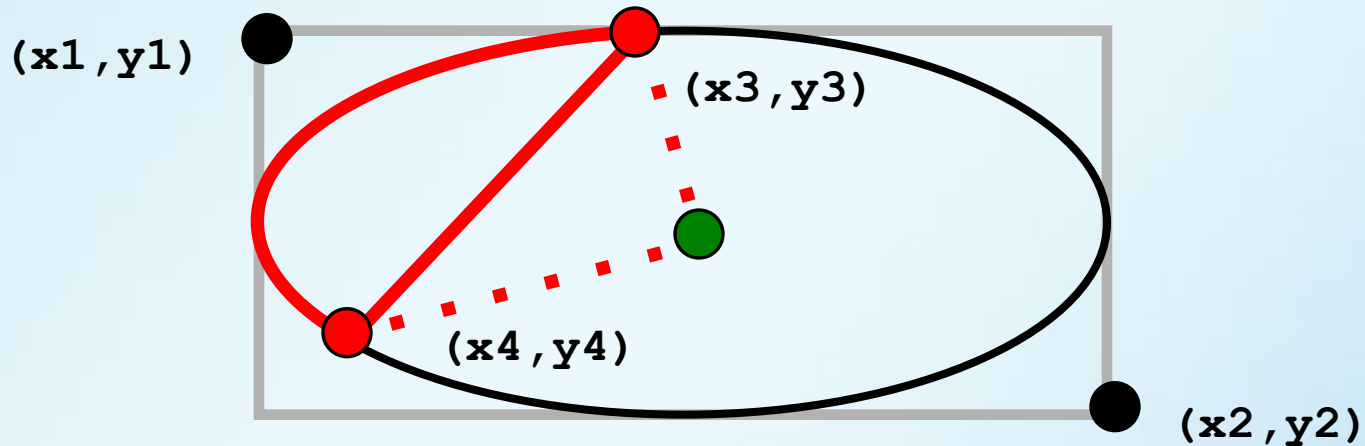
Процедуры рисования

Сегмент эллипса

```
ires = Chord(hdc, x1, y1, x2, y2, x3, y3, x4, y4)
```

Сектор эллипса

```
ires = Pie(hdc, x1, y1, x2, y2, x3, y3, x4, y4)
```



Многоугольник

```
ires = Polygon(hdc, pt, N)
```

type (point) pt поля x, y

N – число вершин

Типы проектов для GDI

Windowing Application

Отображение на экране, в памяти, метафайлах.

Модули `ifwin`, `ifwina`.

Создание оконного приложения.

Функции `WinMain`, `WndProc`.

Console Application

Отсутствует возможность отображения на экране.

Модули `ifwin`, `ifwina`.

Без использования функций `WinMain`, `WndProc`.

Windowing Application

Функции **WinMain** и **WndProc**

WinMain выполняет:

- определение класса окна
- регистрация класса
- создание окна
- отображение окна
- запуск цикла обработки сообщений

MainWndProc обрабатывает поступающие сообщения

WM_CREATE

WM_COMMAND

WM_PAINT

WM_CLOSE

...

WinMain (1)

```
use ifwin      !***** основная программа *****
interface
  integer(4) FUNCTION WinMain(hInstance, hPrevInstance, &
                              lpzCmdLine, nCmdShow)
    !MS$ATTRIBUTES STDCALL, ALIAS : '_WinMain@16' :: WinMain
  integer(4) hInstance, hPrevInstance, lpzCmdLine, nCmdShow
end function WinMain
end interface
end

!----- внешняя функция WinMain -----
integer(4) FUNCTION WinMain(hInstance, hPrevInstance, &
                              lpCmdLine, nCmdShow)
  !MS$ ATTRIBUTES STDCALL, ALIAS : '_WinMain@16' :: WinMain
use ifwina
interface
integer(4) function MainWndProc (hWnd, message, wParam,
lParam)
!MS$ ATTRIBUTES STDCALL, ALIAS : '_MainWndProc@16' ::
MainWndProc
integer hWnd, message, wParam, lParam
end function MainWndProc
```

WinMain (2)

```
!----- формальные параметры -----  
integer hInstance, hPrevInstance, lpCmdLine, nCmdShow  
!----- внутренние константы/переменные -----  
character(50) NameClass /"GDI"C/  
character(100) NameMainWin /"Используем GDI"C/  
logical bret  
integer iret, hWnd  
type (T_MSG) message  
type (T_WNDCLASS) wc  
if (hPrevInstance == 0) then  
    call ZeroMemory (LOC(wc), sizeof(wc)) ! обнуление структуры  
    wc.lpfWndProc = LOC(MainWndProc) ! адрес оконной функции  
    wc.hInstance = hInstance ! дескриптор данного приложения  
    wc.hIcon = LoadIcon(hInstance, IDI_APPLICATION) ! значка  
    wc.hCursor = LoadCursor(NULL, IDC_ARROW) ! курсора  
    wc.hbrBackground = 6 ! цвет фона окна  
    wc.lpszClassName = LOC(NameClass) ! имя класса окна  
    if (RegisterClass (wc) == 0 ) then ! регистрация окна  
        WinMain = FALSE  
        return  
    end if  
end if  
end if
```

WinMain (3)

```
hWnd = CreateWindow(NameClass,          & ! имя класса окна
                   NameMainWin,       & ! имя окна
                   INT(WS_OVERLAPPEDWINDOW), & ! стиль окна
                   0,0,                 & ! верхний левый угол
                   800,600,            & ! размеры окна
                   NULL,               & ! дескриптор родительского
окна
                   NULL,              & ! дескриптор главного меню
                   hInstance, & ! дескриптор приложения
                   NULL) ! указатель на структуру с доп. инф.

nCmdShow = SW_SHOWMAXIMIZED
bret = ShowWindow(hWnd,nCmdShow) ! окно в развернутом виде
bret = UpdateWindow(hWnd)       ! перерисовка рабочей области

do while (GetMessage(message, NULL, 0, 0)) ! обработка сообщений
    bret = TranslateMessage (message)
    iret = DispatchMessage (message)
end do
WinMain = message.wParam
return
end
```

MainWndProc

```
!----- внешняя MainWndProc -----  
integer(4) FUNCTION MainWndProc(hwnd, message, wParam, lParam)  
!MS$  ATTRIBUTES  STDCALL,  ALIAS  :  '_MainWndProc@16'  ::  
MainWndProc  
use ifwin  
integer(4) hwnd, message, wParam, lParam, iret, hdc  
type (T_PAINTSTRUCT) ps  
logical bret  
SELECT CASE (message)  
  CASE (WM_PAINT)  
    hdc = BeginPaint(hwnd,ps)  
    CALL Draw(hdc)  
    bret = EndPaint(hwnd,ps)  
  CASE (WM_DESTROY)  
    call PostQuitMessage(0)  
    MainWndProc = 0  
    return  
  CASE DEFAULT  
    MainWndProc = DefWindowProc(hwnd,message,wParam,lParam)  
    return  
END SELECT  
MainWndProc = 0
```

ВЫЗОВ GDI
процедур

end

Подпрограмма Draw

```
subroutine Draw(hdc)
```

```
  use ifwina
```

```
  integer hdc
```

```
  integer HGREEN_BRUSH, HBLUE_BRUSH
```

```
  integer HPEN, HOLD_PEN, HOLD_BRUSH, ires
```

```
  HBLUE_BRUSH = CreateSolidBrush( RGB(0,0,255) ) ! создание
```

```
  HGREEN_BRUSH = CreateSolidBrush( RGB(0,255,0) )
```

```
  HPEN=CreatePen( PS_SOLID,10,RGB(230,60,150) )
```

```
  HOLD_PEN = SelectObject(hdc,HPEN) ! сохранение
```

```
  HOLD_BRUSH = SelectObject(hdc,HGREEN_BRUSH)
```

```
  ires = Rectangle(hdc,100,200,400,300)
```

```
  ires = SelectObject(hdc,HOLD_PEN) ! восстановление
```

```
  ires = SelectObject(hdc,hBLUE_BRUSH)
```

```
  ires = Ellipse(hdc,150,150,250,250)
```

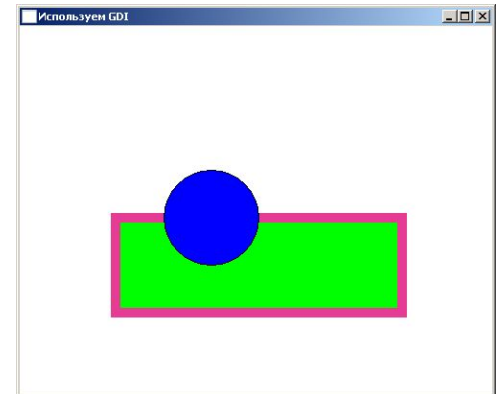
```
  ires = SelectObject(hdc,HOLD_BRUSH)
```

```
  ires = DeleteObject(HPEN)
```

```
  ires = DeleteObject(HBLUE_BRUSH)
```

```
  ires = DeleteObject(HGREEN_BRUSH)
```

```
end subroutine Draw
```



Console Application

Графический вывод в метафайл

```
program DrawToMetaFile
  use ifwina
  integer hEMF, hBRUSH, hPEN, ires

  hEMF    = CreateEnhMetaFile (0, "D:\\pic.emf", null_rect, "A")

  hBRUSH  = CreateSolidBrush (RGB (0, 255, 0))
  hPEN    = CreatePen (PS_DASH, 4, Rgb (255, 0, 0))

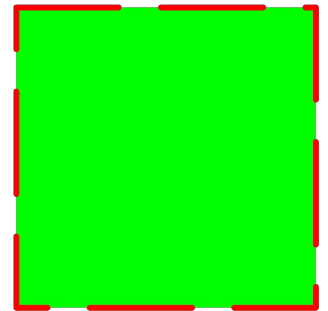
  ires = SelectObject (hEMF, hBRUSH)
  ires = SelectObject (hEMF, hPEN)

  ires = Rectangle (hEMF, 100, 100, 300, 300)

  ires = CloseEnhMetaFile (hEMF)

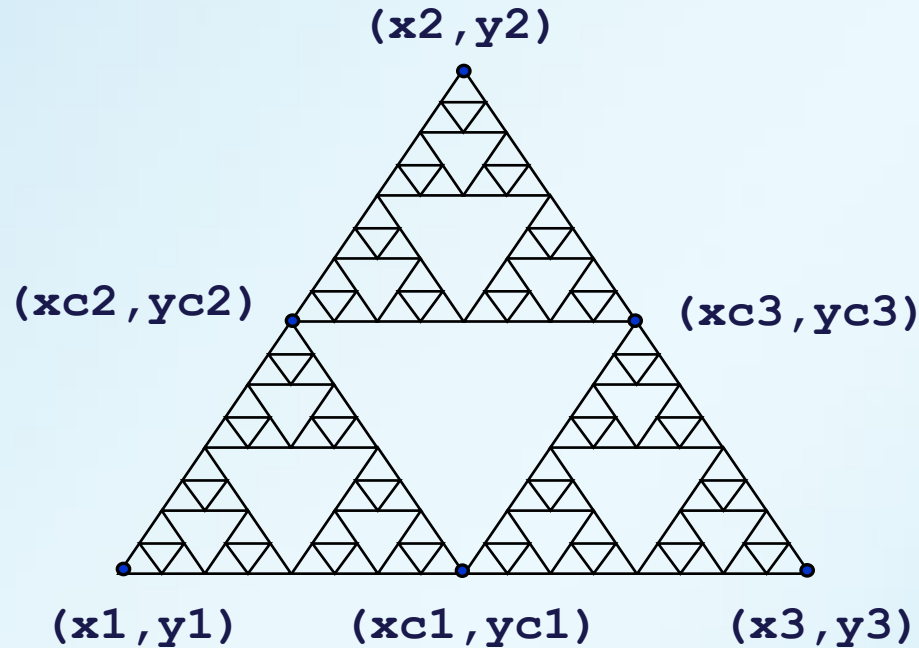
  ires = DeleteObject (HPEN)
  ires = DeleteObject (HBRUSH)

end
```



* З а д а н и е *

Рисование треугольника Серпинского.



Построить главный треугольник.

Найти середины сторон.

Сообщить трем треугольникам-потомкам,
проделать выше-сказанное.

(рекурсивный вызов)

* Вариант программы *

```
!*****  
!           Рисование треугольника Серпинского в метафайл  
!*****  
program FRACTAL  
  use ifwina  
  implicit none  
  integer hEMF, hPEN, ires  
  hEMF = CreateEnhMetaFile (0,"D:\\Serpinsky.emf"C,null_rect,""C) ! --- создание метафайла, пера  
  hPEN = CreatePen (PS_SOLID,1,Rgb(0,0,255))  
  ires = SelectObject(hEMF, hPEN)  
  
  call Serpinsky(hEMF, 50, 200, 150, 50, 250, 200, 5)! --- вызов рекурсивной подпрограммы рисования  
  
  ires=CloseEnhMetaFile(hEMF)  
  ires=DeleteObject(hPEN)  
contains  
  recursive subroutine Serpinsky(hc,x1,y1,x2,y2,x3,y3,N)  
    integer hc ! дескриптор метафайла  
    integer x1,y1,x2,y2,x3,y3 ! координаты треугольника  
    integer N ! число поколений  
    integer xc1, yc1, xc2, yc2, xc3, yc3! координаты середин сторон  
    integer ires  
  
    if (N==0) return ! остановка рекурсии  
  
    ires=MoveToEx(hc, x1, y1, NULL)  
    ires=LineTo(hc, x2, y2)  
    ires=LineTo(hc, x3, y3)  
    ires=LineTo(hc, x1, y1)  
    xc1=(x1+x3)/2; yc1=(y1+y3)/2  
    xc2=(x1+x2)/2; yc2=(y1+y2)/2  
    xc3=(x2+x3)/2; yc3=(y2+y3)/2  
  
    call Serpinsky(hc,x1,y1,xc2,yc2,xc1,yc1,N-1)  
    call Serpinsky(hc,xc2,yc2,x2,y2,xc3,yc3,N-1)  
    call Serpinsky(hc,xc1,yc1,xc3,yc3,x3,y3,N-1)  
  end subroutine Serpinsky  
end
```