Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Институт информационных технологий

Факультет повышения квалификации и переподготовки

Кафедра МПСС

**ОТЧЕТ**

По самостоятельной работе №4

по дисциплине ТКП

Тема: «Разработка элемент управления ActiveX на базе библиотеки ATL»

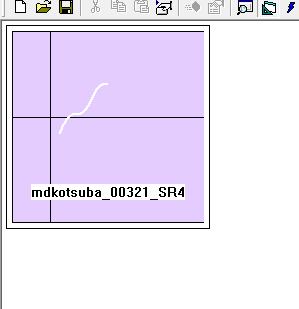
Минск 2021

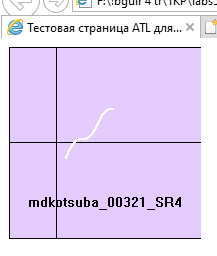
**Цель работы:**

Разработать элемент управления ActiveX на базе библиотеки ATL. Элемент управления должен: – выводить график определенной в соответствии с индивидуальным вариантом функции (при этом в исходном состоянии график не отображается); – обеспечивать обработку соответствующих щелчкам левой и правой кнопок мыши событий “LButton” и “RButton”; – поддерживать свойства: 1) цвет графика; 2) цвет фона; 3) минимальное и максимальное значения координат X и Y, определяющих протяженность и положение на координатной плоскости области вывода графика функции. События элемента управления: – «LButton». При щелчке левой кнопкой появляется отображение графика. Повторный щелчок (не после щелчка правой кнопкой) приводит к генерации ошибки; – «RButton». При щелчке правой кнопки мыши изображение графика исчезает. Повторный (при отсутствующем изображении графика) щелчок правой кнопкой приводит к генерации ошибки.

**Индивидуальное задание:**







Проект ***Graf***

|  |
| --- |
| // Graf.cpp: реализация экспортов DLL.  #include "pch.h"  #include "framework.h"  #include "resource.h"  #include "Graf\_i.h"  #include "dllmain.h"  using namespace ATL;  // Используется, чтобы определить, можно ли выгрузить DLL средствами OLE.  \_Use\_decl\_annotations\_  STDAPI DllCanUnloadNow(void)  {  return \_AtlModule.DllCanUnloadNow();  }  // Возвращает фабрику класса для создания объекта требуемого типа.  \_Use\_decl\_annotations\_  STDAPI DllGetClassObject(\_In\_ REFCLSID rclsid, \_In\_ REFIID riid, \_Outptr\_ LPVOID\* ppv)  {  return \_AtlModule.DllGetClassObject(rclsid, riid, ppv);  }  // DllRegisterServer - добавляет записи в системный реестр.  \_Use\_decl\_annotations\_  STDAPI DllRegisterServer(void)  {  // регистрирует объект, библиотеку типов и все интерфейсы в библиотеке типов  HRESULT hr = \_AtlModule.DllRegisterServer();  return hr;  }  // DllUnregisterServer - удаляет записи из системного реестра.  \_Use\_decl\_annotations\_  STDAPI DllUnregisterServer(void)  {  HRESULT hr = \_AtlModule.DllUnregisterServer();  return hr;  }  // DllInstall - добавляет и удаляет записи системного реестра для конкретного пользователя и компьютера.  STDAPI DllInstall(BOOL bInstall, \_In\_opt\_ LPCWSTR pszCmdLine)  {  HRESULT hr = E\_FAIL;  static const wchar\_t szUserSwitch[] = L"user";  if (pszCmdLine != nullptr)  {  if (\_wcsnicmp(pszCmdLine, szUserSwitch, \_countof(szUserSwitch)) == 0)  {  ATL::AtlSetPerUserRegistration(true);  }  }  if (bInstall)  {  hr = DllRegisterServer();  if (FAILED(hr))  {  DllUnregisterServer();  }  }  else  {  hr = DllUnregisterServer();  }  return hr;  } |

|  |
| --- |
| // Graf.idl: источник языка IDL для Graf  //  // Этот файл будет обработан средством MIDL для  // создание библиотеки типов ($handlersafeidlname$.tlb) и кода маршалирования.  #include "olectl.h"  import "oaidl.idl";  import "ocidl.idl";  [  object,  uuid(cff968f4-7aa6-4d49-8b0a-2c9629509681),  dual,  nonextensible,  pointer\_default(unique)  ]  interface IGrafCtl : IDispatch  {  [propput, bindable, requestedit, id(DISPID\_BACKCOLOR)]  HRESULT BackColor([in]OLE\_COLOR clr);  [propget, bindable, requestedit, id(DISPID\_BACKCOLOR)]  HRESULT BackColor([out,retval]OLE\_COLOR\* pclr);  [propput, bindable, requestedit, id(DISPID\_BORDERSTYLE)]  HRESULT BorderStyle([in]long style);  [propget, bindable, requestedit, id(DISPID\_BORDERSTYLE)]  HRESULT BorderStyle([out, retval]long\* pstyle);  [propput, bindable, requestedit, id(DISPID\_FILLCOLOR)]  HRESULT FillColor([in]OLE\_COLOR clr);  [propget, bindable, requestedit, id(DISPID\_FILLCOLOR)]  HRESULT FillColor([out, retval]OLE\_COLOR\* pclr);  [propget, id(1), helpstring("Начало по оси X")] HRESULT StartX([out, retval] LONG\* pVal);  [propput, id(1), helpstring("Начало по оси X")] HRESULT StartX([in] LONG newVal);  [propget, id(2), helpstring("Конец по оси X")] HRESULT EndX([out, retval] LONG\* pVal);  [propput, id(2), helpstring("Конец по оси X")] HRESULT EndX([in] LONG newVal);  [propget, id(3), helpstring("Начало по оси Y")] HRESULT StartY([out, retval] LONG\* pVal);  [propput, id(3), helpstring("Начало по оси Y")] HRESULT StartY([in] LONG newVal);  [propget, id(4), helpstring("Конец по оси Y")] HRESULT EndY([out, retval] LONG\* pVal);  [propput, id(4), helpstring("Конец по оси Y")] HRESULT EndY([in] LONG newVal);  };  [  uuid(58fe36b6-a0c7-447e-b1bb-f780c5775a7f),  version(1.0),  ]  library GrafLib  {  importlib("stdole2.tlb");  [  uuid(08bc11d6-2f27-44c4-8baa-60e3799fb18a)  ]  dispinterface \_IGrafCtlEvents  {  properties:  methods:  [id(1)] HRESULT ClickL();  [id(2)] HRESULT ClickR();  };  [  uuid(620ea895-635b-4fb2-8ddd-0e0bdf234e03),  control  ]  coclass GrafCtl  {  [default] interface IGrafCtl;  [default, source] dispinterface \_IGrafCtlEvents;  };  }; |

|  |
| --- |
| // GrafCtl.cpp: реализация CGrafCtl  #include "pch.h"  #include "GrafCtl.h"  // CGrafCtl  STDMETHODIMP CGrafCtl::get\_StartX(LONG\* pVal)  {  // TODO: Добавьте сюда код реализации  \*pVal = m\_nStartX;  return S\_OK;  }  STDMETHODIMP CGrafCtl::put\_StartX(LONG newVal)  {  // TODO: Добавьте сюда код реализации  m\_nStartX = newVal;  FireViewChange();  return S\_OK;  }  void CGrafCtl::CalcPoints(const RECT& rc)  {  double x;  x = m\_nStartX;  double scaleX = (rc.right - rc.left) / 20.;// (1 \* (\_xEnd - \_xStart));  double scaleY = (rc.bottom - rc.top) / 20.;// / (\_yEnd - \_yStart);  double offsetX = 5 \* scaleX;  double offsetY = (rc.bottom - rc.top) / 2.;  for (int i = 0; i < nPoints; i++)  {  m\_arrPoint[i].x = (long)(((x + 2) \* scaleX) + offsetX);  m\_arrPoint[i].y = -(long)((x - pow(sin(x), 2)) \* scaleY - offsetY);  x += ((double)(m\_nEndX - m\_nStartX)) / (nPoints - 1);  }  }  STDMETHODIMP CGrafCtl::get\_EndX(LONG\* pVal)  {  // TODO: Добавьте сюда код реализации  \*pVal = m\_nEndX;  return S\_OK;  }  STDMETHODIMP CGrafCtl::put\_EndX(LONG newVal)  {  // TODO: Добавьте сюда код реализации  m\_nEndX = newVal;  FireViewChange();  return S\_OK;  }  STDMETHODIMP CGrafCtl::get\_StartY(LONG\* pVal)  {  // TODO: Добавьте сюда код реализации  \*pVal = m\_nStartY;  return S\_OK;  }  STDMETHODIMP CGrafCtl::put\_StartY(LONG newVal)  {  // TODO: Добавьте сюда код реализации  m\_nStartY = newVal;  FireViewChange();  return S\_OK;  }  STDMETHODIMP CGrafCtl::get\_EndY(LONG\* pVal)  {  // TODO: Добавьте сюда код реализации  \*pVal = m\_nEndY;  return S\_OK;  }  STDMETHODIMP CGrafCtl::put\_EndY(LONG newVal)  {  // TODO: Добавьте сюда код реализации  m\_nEndY = newVal;  FireViewChange();  return S\_OK;  }  LRESULT CGrafCtl::OnLButtonDown(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)  {    if (Error == true)  {  ClicOnOff = true;  FireViewChange();  Error = False;  return 0;  }  else  {  Error = true;  FireViewChange();  return 0;  }    }  LRESULT CGrafCtl::OnRButtonDown(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)  {  if (Error == False)  {  ClicOnOff = False;  FireViewChange();  Error = true;  return 0;  }  else  {  Error = False;  FireViewChange();  return 0;  }  return 0;  } |

|  |
| --- |
| #pragma once  using namespace ATL;  template <class T>  class CProxy\_IGrafCtlEvents : public IConnectionPointImpl<T, &\_\_uuidof(\_IGrafCtlEvents), CComDynamicUnkArray>  {  // ПРЕДУПРЕЖДЕНИЕ: Мастер может повторить создание этого класса  public:  HRESULT Fire\_ClickIn(LONG x, LONG y)  {  HRESULT hr = S\_OK;  T\* pThis = static\_cast<T\*>(this);  int cConnections = m\_vec.GetSize();  for (int iConnection = 0; iConnection < cConnections; iConnection++)  {  pThis->Lock();  CComPtr<IUnknown> punkConnection = m\_vec.GetAt(iConnection);  pThis->Unlock();  IDispatch\* pConnection = static\_cast<IDispatch\*>(punkConnection.p);  if (pConnection)  {  CComVariant avarParams[2];  avarParams[1] = x;  avarParams[1].vt = VT\_I4;  avarParams[0] = y;  avarParams[0].vt = VT\_I4;  DISPPARAMS params = { avarParams, NULL, 2, 0 };  hr = pConnection->Invoke(1, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_METHOD, &params, NULL, NULL, NULL);  }  }  return hr;  }  HRESULT Fure\_ClickOut(LONG x, LONG y)  {  HRESULT hr = S\_OK;  T\* pThis = static\_cast<T\*>(this);  int cConnections = m\_vec.GetSize();  for (int iConnection = 0; iConnection < cConnections; iConnection++)  {  pThis->Lock();  CComPtr<IUnknown> punkConnection = m\_vec.GetAt(iConnection);  pThis->Unlock();  IDispatch\* pConnection = static\_cast<IDispatch\*>(punkConnection.p);  if (pConnection)  {  CComVariant avarParams[2];  avarParams[1] = x;  avarParams[1].vt = VT\_I4;  avarParams[0] = y;  avarParams[0].vt = VT\_I4;  DISPPARAMS params = { avarParams, NULL, 2, 0 };  hr = pConnection->Invoke(2, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_METHOD, &params, NULL, NULL, NULL);  }  }  return hr;  }  }; |

|  |
| --- |
| // GrafCtl.h: объявление CGrafCtl  #pragma once  #include "resource.h" // основные символы  #include <atlctl.h>  #include "Graf\_i.h"  #include "\_IGrafCtlEvents\_CP.h"  #include <math.h>  #if defined(\_WIN32\_WCE) && !defined(\_CE\_DCOM) && !defined(\_CE\_ALLOW\_SINGLE\_THREADED\_OBJECTS\_IN\_MTA)  #error "Однопотоковые COM-объекты не поддерживаются должным образом платформой Windows CE, например платформами Windows Mobile, в которых не предусмотрена полная поддержка DCOM. Определите \_CE\_ALLOW\_SINGLE\_THREADED\_OBJECTS\_IN\_MTA, чтобы принудить ATL поддерживать создание однопотоковых COM-объектов и разрешить использование его реализаций однопотоковых COM-объектов. Для потоковой модели в вашем rgs-файле задано значение 'Free', поскольку это единственная потоковая модель, поддерживаемая не-DCOM платформами Windows CE."  #endif  using namespace ATL;  // CGrafCtl  class ATL\_NO\_VTABLE CGrafCtl :  public CComObjectRootEx<CComSingleThreadModel>,  public CStockPropImpl<CGrafCtl, IGrafCtl>,  public IPersistStreamInitImpl<CGrafCtl>,  public IOleControlImpl<CGrafCtl>,  public IOleObjectImpl<CGrafCtl>,  public IOleInPlaceActiveObjectImpl<CGrafCtl>,  public IViewObjectExImpl<CGrafCtl>,  public IOleInPlaceObjectWindowlessImpl<CGrafCtl>,  public ISupportErrorInfo,  public IConnectionPointContainerImpl<CGrafCtl>,  public CProxy\_IGrafCtlEvents<CGrafCtl>,  public IObjectWithSiteImpl<CGrafCtl>,  public IServiceProviderImpl<CGrafCtl>,  public IPersistStorageImpl<CGrafCtl>,  public ISpecifyPropertyPagesImpl<CGrafCtl>,  public IQuickActivateImpl<CGrafCtl>,  #ifndef \_WIN32\_WCE  public IDataObjectImpl<CGrafCtl>,  #endif  public IProvideClassInfo2Impl<&CLSID\_GrafCtl, &\_\_uuidof(\_IGrafCtlEvents), &LIBID\_GrafLib>,  public IPropertyNotifySinkCP<CGrafCtl>,  public IObjectSafetyImpl<CGrafCtl, INTERFACESAFE\_FOR\_UNTRUSTED\_CALLER>,  public CComCoClass<CGrafCtl, &CLSID\_GrafCtl>,  public CComControl<CGrafCtl>  {  public:  CGrafCtl()  {  m\_nStartX = -1;  m\_nEndX = 4;  m\_nStartY = -1;  m\_nEndY = 4;  m\_clrFillColor = RGB(229,204,255);  m\_clrBackColor = RGB(51, 0, 102);    }  DECLARE\_OLEMISC\_STATUS(OLEMISC\_RECOMPOSEONRESIZE |  OLEMISC\_CANTLINKINSIDE |  OLEMISC\_INSIDEOUT |  OLEMISC\_ACTIVATEWHENVISIBLE |  OLEMISC\_SETCLIENTSITEFIRST  )  DECLARE\_REGISTRY\_RESOURCEID(IDR\_GRAFCTL)  BEGIN\_COM\_MAP(CGrafCtl)  COM\_INTERFACE\_ENTRY(IGrafCtl)  COM\_INTERFACE\_ENTRY(IDispatch)  COM\_INTERFACE\_ENTRY(IViewObjectEx)  COM\_INTERFACE\_ENTRY(IViewObject2)  COM\_INTERFACE\_ENTRY(IViewObject)  COM\_INTERFACE\_ENTRY(IOleInPlaceObjectWindowless)  COM\_INTERFACE\_ENTRY(IOleInPlaceObject)  COM\_INTERFACE\_ENTRY2(IOleWindow, IOleInPlaceObjectWindowless)  COM\_INTERFACE\_ENTRY(IOleInPlaceActiveObject)  COM\_INTERFACE\_ENTRY(IOleControl)  COM\_INTERFACE\_ENTRY(IOleObject)  COM\_INTERFACE\_ENTRY(IPersistStreamInit)  COM\_INTERFACE\_ENTRY2(IPersist, IPersistStreamInit)  COM\_INTERFACE\_ENTRY(ISupportErrorInfo)  COM\_INTERFACE\_ENTRY(IConnectionPointContainer)  COM\_INTERFACE\_ENTRY(ISpecifyPropertyPages)  COM\_INTERFACE\_ENTRY(IQuickActivate)  COM\_INTERFACE\_ENTRY(IPersistStorage)  #ifndef \_WIN32\_WCE  COM\_INTERFACE\_ENTRY(IDataObject)  #endif  COM\_INTERFACE\_ENTRY(IProvideClassInfo)  COM\_INTERFACE\_ENTRY(IProvideClassInfo2)  COM\_INTERFACE\_ENTRY(IObjectWithSite)  COM\_INTERFACE\_ENTRY(IServiceProvider)  COM\_INTERFACE\_ENTRY\_IID(IID\_IObjectSafety, IObjectSafety)  END\_COM\_MAP()  BEGIN\_PROP\_MAP(CGrafCtl)  PROP\_DATA\_ENTRY("\_cx", m\_sizeExtent.cx, VT\_UI4)  PROP\_DATA\_ENTRY("\_cy", m\_sizeExtent.cy, VT\_UI4)  #ifndef \_WIN32\_WCE  PROP\_ENTRY\_TYPE("BackColor", DISPID\_BACKCOLOR, CLSID\_StockColorPage, VT\_UI4)  #endif  PROP\_ENTRY\_TYPE("BorderStyle", DISPID\_BORDERSTYLE, CLSID\_NULL, VT\_I4)  #ifndef \_WIN32\_WCE  PROP\_ENTRY\_TYPE("FillColor", DISPID\_FILLCOLOR, CLSID\_StockColorPage, VT\_UI4)  #endif  // Записи примеров  // PROP\_ENTRY\_TYPE("Имя свойства", dispid, clsid, vtType)  // PROP\_PAGE(CLSID\_StockColorPage)  END\_PROP\_MAP()  BEGIN\_CONNECTION\_POINT\_MAP(CGrafCtl)  CONNECTION\_POINT\_ENTRY(IID\_IPropertyNotifySink)  CONNECTION\_POINT\_ENTRY(\_\_uuidof(\_IGrafCtlEvents))  END\_CONNECTION\_POINT\_MAP()  BEGIN\_MSG\_MAP(CGrafCtl)  CHAIN\_MSG\_MAP(CComControl<CGrafCtl>)  DEFAULT\_REFLECTION\_HANDLER()  MESSAGE\_HANDLER(WM\_LBUTTONDOWN, OnLButtonDown);  MESSAGE\_HANDLER(WM\_RBUTTONDOWN, OnRButtonDown);  END\_MSG\_MAP()  // Прототипы обработчика:  // LRESULT MessageHandler(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);  // LRESULT CommandHandler(WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);  // LRESULT NotifyHandler(int idCtrl, LPNMHDR pnmh, BOOL& bHandled);  LRESULT OnLButtonDown(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);  LRESULT OnRButtonDown(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);  // ISupportsErrorInfo  STDMETHOD(InterfaceSupportsErrorInfo)(REFIID riid)  {  static const IID\* const arr[] =  {  &IID\_IGrafCtl,  };  for (int i=0; i<sizeof(arr)/sizeof(arr[0]); i++)  {  if (InlineIsEqualGUID(\*arr[i], riid))  return S\_OK;  }  return S\_FALSE;  }  // IViewObjectEx  DECLARE\_VIEW\_STATUS(VIEWSTATUS\_SOLIDBKGND | VIEWSTATUS\_OPAQUE)  // IGrafCtl  static const int nPoints = 100;  POINT m\_arrPoint[nPoints];  void CalcPoints(const RECT& rc);  bool ClicOnOff;  bool Error = True;  public:  HRESULT OnDraw(ATL\_DRAWINFO& di)  {  COLORREF colFore;  HBRUSH hOldBrush, hBrush;  HPEN hOldPen, hPen;  // Translate m\_colFore into a COLORREF type  OleTranslateColor(m\_clrFillColor, NULL, &colFore);  hBrush = CreateSolidBrush(colFore);  SelectObject(di.hdcDraw, hBrush);  RECT& rc = \*(RECT\*)di.prcBounds;  // Ограничивает область вырезания прямоугольником, указанным в di.prcBounds  HRGN hRgnOld = nullptr;  if (GetClipRgn(di.hdcDraw, hRgnOld) != 1)  hRgnOld = nullptr;  bool bSelectOldRgn = false;  HRGN hRgnNew = CreateRectRgn(rc.left, rc.top, rc.right, rc.bottom);  if (hRgnNew != nullptr)  {  bSelectOldRgn = (SelectClipRgn(di.hdcDraw, hRgnNew) != ERROR);  }  Rectangle(di.hdcDraw, rc.left, rc.top, rc.right \* 2, rc.bottom);  // ///////////////////////////////////////////////////////////////  HDC hdc = di.hdcDraw;  // Create and select the colors to draw the circle  hPen = (HPEN)GetStockObject(BLACK\_PEN);  hOldPen = (HPEN)SelectObject(hdc, hPen);  hBrush = (HBRUSH)GetStockObject(WHITE\_BRUSH);  hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);  // Create and select the brush that will be used to fill the graf  //hBrush = CreateSolidBrush(colFore);  //SelectObject(hdc, hBrush);  CalcPoints(rc);  MoveToEx(hdc, (rc.right - rc.left) / 4, rc.top, NULL);  LineTo(hdc, (rc.right - rc.left) / 4, rc.bottom);  MoveToEx(hdc, rc.left, (rc.bottom - rc.top) / 2, NULL);  LineTo(hdc, rc.right, (rc.bottom - rc.top) / 2);  hBrush = CreateSolidBrush(m\_clrFillColor);  SelectObject(hdc, hBrush);  hPen = CreatePen(PS\_SOLID, 2, m\_clrBackColor);  SelectObject(hdc, hPen);  if (ClicOnOff == true)  {  Polyline(hdc, &m\_arrPoint[0], nPoints);  }    // Select back the old pen and brush and delete the brush we created  SelectObject(hdc, hOldPen);  //SelectObject(hdc, hOldBrush);  //DeleteObject(hBrush);  DeleteObject(hPen);  //TextOut()    SetTextAlign(di.hdcDraw, TA\_CENTER | TA\_BASELINE);  LPCTSTR pszText = \_T("mdkotsuba\_00321\_SR4");  //#ifndef \_WIN32\_WCE  TextOut(di.hdcDraw,  (rc.left + rc.right) / 2,  (rc.top + rc.bottom) / 1.2,  pszText,  lstrlen(pszText));  /\*--------------------------------------------------------------------\*/  #ifndef \_WIN32\_WCE    #else  ExtTextOut(di.hdcDraw,  (rc.left + rc.right) / 2,  (rc.top + rc.bottom) / 2,  ETO\_OPAQUE,  nullptr,  pszText,  ATL::lstrlen(pszText),  nullptr);  #endif  if (bSelectOldRgn)  SelectClipRgn(di.hdcDraw, hRgnOld);  DeleteObject(hRgnNew);  return S\_OK;  }  OLE\_COLOR m\_clrBackColor;  void OnBackColorChanged()  {  ATLTRACE(\_T("OnBackColorChanged\n"));  }  LONG m\_nBorderStyle;  void OnBorderStyleChanged()  {  ATLTRACE(\_T("OnBorderStyleChanged\n"));  }  OLE\_COLOR m\_clrFillColor;  void OnFillColorChanged()  {  ATLTRACE(\_T("OnFillColorChanged\n"));  }  STDMETHOD(\_InternalQueryService)(REFGUID /\*guidService\*/, REFIID /\*riid\*/, void\*\* /\*ppvObject\*/)  {  return E\_NOTIMPL;  }  DECLARE\_PROTECT\_FINAL\_CONSTRUCT()  HRESULT FinalConstruct()  {  return S\_OK;  }  void FinalRelease()  {  }  INT16 m\_nStartX;  INT16 m\_nEndX;  INT16 m\_nStartY;  INT16 m\_nEndY;  STDMETHOD(get\_StartX)(LONG\* pVal);  STDMETHOD(put\_StartX)(LONG newVal);  STDMETHOD(get\_EndX)(LONG\* pVal);  STDMETHOD(put\_EndX)(LONG newVal);  STDMETHOD(get\_StartY)(LONG\* pVal);  STDMETHOD(put\_StartY)(LONG newVal);  STDMETHOD(get\_EndY)(LONG\* pVal);  STDMETHOD(put\_EndY)(LONG newVal);  };  OBJECT\_ENTRY\_AUTO(\_\_uuidof(GrafCtl), CGrafCtl) |