

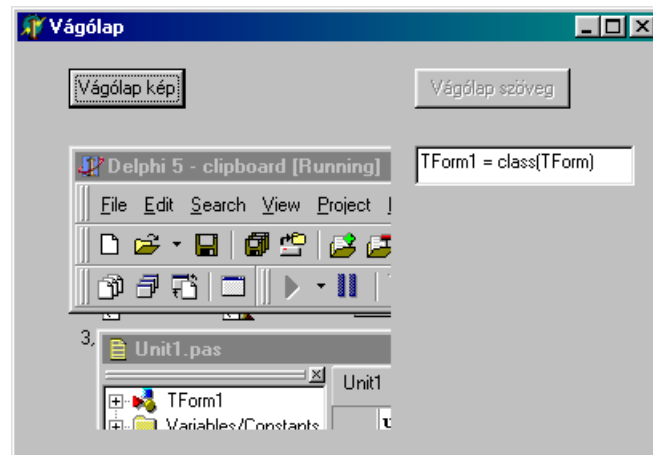
### 8.3 Alkalmazások közötti kapcsolatok

1. Vágólapnéző [Clipboard](#)
2. OLE-konténer használata [OLECont](#)
3. A Word elérése OLE automatizmussal [WordAuto](#)
4. Az Excel és az OLE automatizmus [ExcelAuto](#)
5. A Word elérése OLEServerként [WordServer](#)
6. Az Excel használata OLEServerként [ExcelServer](#)



Készítsünk programot, amely egy *Image*, illetve egy *Edit* komponensbe másolja a vágólap tartalmát, attól függően, hogy mi van a vágólapon! (*Clipboard*)

Az alkalmazás ablaka mindössze négy építőelemet tartalmaz (két nyomógombot, egy képvezérlőt és egy szövegmezőt):



Ahhoz, hogy a vágólapot használjuk, be kell építeni a *clipbrd* modult a programunkba:

```
uses clipbrd;
```

Ha mozgatjuk az egeret az ablakon a vágólap tartalmának megfelelően aktivizálhatók a lehívó gombok:

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState;  
    X, Y: Integer);  
begin  
    // Mi van a vágólapon  
    Button1.Enabled:=Clipboard.HasFormat(CF_PICTURE) or  
        Clipboard.HasFormat(CF_BITMAP) or  
        Clipboard.HasFormat(CF_METAFILEPICT);  
    Button2.Enabled:=Clipboard.HasFormat(CF_TEXT);  
end;
```

A szöveget az *Edit1* vezérlőbe másoljuk:

```
procedure TForm1.Button2Click(Sender: TObject);  
begin  
    Edit1.Text:=Clipboard.AsText;  
end;
```

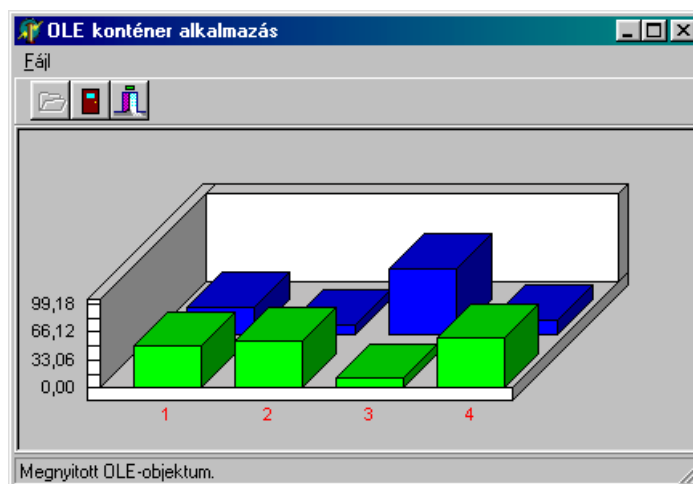
A képet az *Image1* vezérlőbe töltjük:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    with Clipboard do  
        try  
            Open;  
            Image1.Picture.Assign(ClipBoard);  
        finally  
            Close;  
        end;  
end;
```



Készítsünk olyan alkalmazást, amelyben egy OLE-konténer bármilyen OLE-kiszolgáló által kezelt objektumot képes magába fogadni! (*OLECont*)

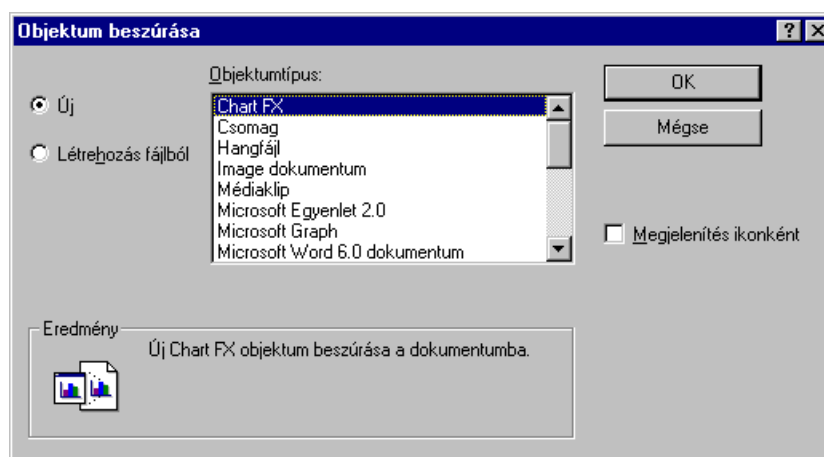
A program futás közbeni ablaka *ChartFx* objektum beszerkesztése esetén:



A megoldás elkészítéséhez egy SDI alkalmazás vázából indulunk ki, melyet a „*File/New/Projects/SDI Application*” választással állíthatunk elő. Az alkalmazás formjára egy panelt (*Panel1*) helyezünk, és ennek *Align* tulajdonságát *alClient* értékre állítjuk. Erre a panelre tesszük az *OleContainer* komponenst szintén *alClient* értékű *Align* tulajdonsággal.

Az SDI alkalmazásban csak a *Fájl* menüt használjuk, *Megnyit*, *Bezár* és *Kilép* menüpontokkal. A menühöz kapcsolódóan a *SpeedPanel1*-en gyorsítógombokat működtetünk. A *SpeedPanel1 Locked* tulajdonságát *true* értékre állítjuk, hogy az OLE-kiszolgáló alkalmazás ne tudja megváltoztatni az eszközsort. A *Statusbar* komponenst szöveg megjelenítésre használjuk.

A tervezés során beállítjuk az *OleContainer* által kezelt objektumot valamelyik OLE-kiszolgáló által kezelt objektumra, hogy a program indításakor is módosítható legyen objektum. Az OLE-szervert a komponens területén belüli kettős kattintás hatására megjelenő párbeszédablakból választhatjuk ki:



Ugyanez a párbeszédablak jelenik meg, a futó alkalmazás *Fájl/Megnyit...* menüpontjának, illetve a megfelelő eszközgomb kiválasztásakor is. Ekkor a megjelenítést az *InsertObjectDialog* metódus hívása végzi. (Indítás, illetve a menüválasztás után ezt a funkciót letiltjuk, így új objektummal csak az előző lezárása után lehet felvenni a kapcsolatot.)

```

procedure TSDIAppForm.FormCreate(Sender: TObject);
begin
    Application.OnHint := ShowHint;
    Megnyit.Enabled:=False;
    MegnyitBtn.Enabled:=False;
end;

procedure TSDIAppForm.OpenItemClick(Sender: TObject);
begin
    OleContainer1.InsertObjectDialog;
    StatusBar.SimpleText:='Megnyitott OLE-objektum.';
    Megnyit.Enabled:=False;
    Bezar.Enabled:=True;
    MegnyitBtn.Enabled:=False;
    BezarBtn.Enabled:=True;
end;

```

A *Bezár* menüpont és eszközgomb használható az OLE-kiszolgáló lezárására és az objektum megsemmisítésére:

```

procedure TSDIAppForm.BezarBtnClick(Sender: TObject);
begin
    OleContainer1.Close;
    OleContainer1.DestroyObject;
    StatusBar.SimpleText:='Nincs megnyitott OLE-objektum.';
    Bezar.Enabled:=False;
    Megnyit.Enabled:=True;
    BezarBtn.Enabled:=False;
    MegnyitBtn.Enabled:=True;
end;

```



Készítsünk programot, mely adott szöveggel, egy új dokumentummal nyitja meg a Word alkalmazást! Lehesse bezárni, és újra megnyitni a Word alkalmazást! Használjuk az OLE automatizmust! (*WordAuto*)

Helyezzünk két gombot a formra! A *Button1* felirata „Megnyitás”, a *Button2* felirata „Bezárás”. Ez utóbbi nem engedélyezett induláskor.

A globálisan deklarált *Variant* típust használjuk.

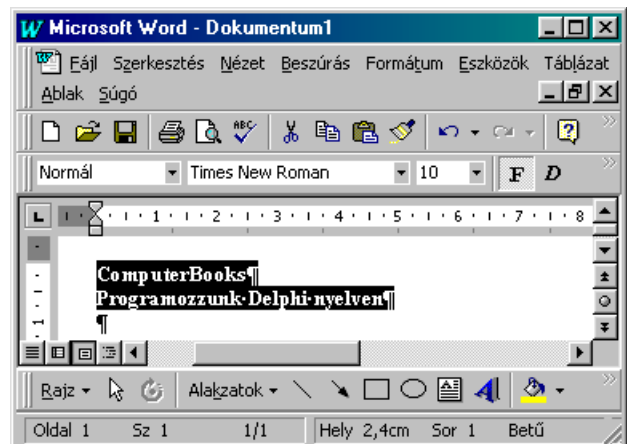
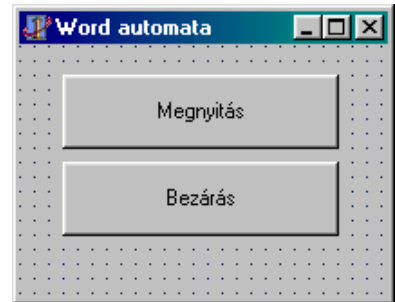
```
uses comobj;  
// Az OLE objektum azonosítója  
var Word:variant;
```

A *Button1* megnyomásakor elindítjuk a kiszolgálót.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    try  
        // Az objektum létrehozása  
        Word:=CreateOleObject('Word.Application');  
        // A Word látható  
        Word.visible:=true;  
        // Új dokumentum nyitása  
        Word.Documents.Add;  
        // betűtípus beállítása és szöveg hozzáadása  
        Word.Selection.Font.Bold:=true;  
        Word.Selection.InsertAfter('ComputerBooks' + #13);  
        Word.Selection.InsertAfter('Programozzuk Delphi nyelven' + #13);  
        Button1.Enabled:=false;  
        Button2.Enabled:=true;  
    except  
    end;  
end;
```

A *Button2* bezárja a Word alkalmazást. A gombok maguk kezelik engedélyezettségüket.

```
procedure TForm1.Button2Click(Sender: TObject);  
begin  
    // A Word lezárása  
    try  
        Word.Quit(0,0,0);  
    except  
    end;  
    Button1.Enabled:=true;  
    Button2.Enabled:=false;  
end;
```



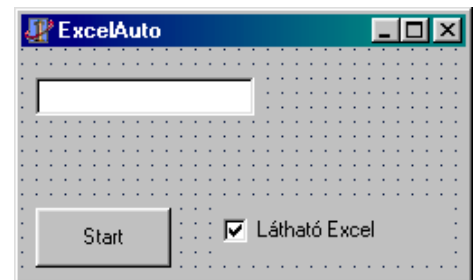


Készítsünk programot, amely meghatározott darabszámú adatot tölt egy Excel-tábla celláiba, az Excel összegzi azokat és az eredményt visszaadja a programnak! Használjuk az OLE automatizmust! (*ExcelAuto*)

Készítsük el a formot! Legyen rajta egy gomb (*Button1*) *Start* felirattal, egy szövegmező (*Edit1*) és egy jelölőnégyzet (*CheckBox1*) annak beállítására, hogy látható-e az *Excel*!

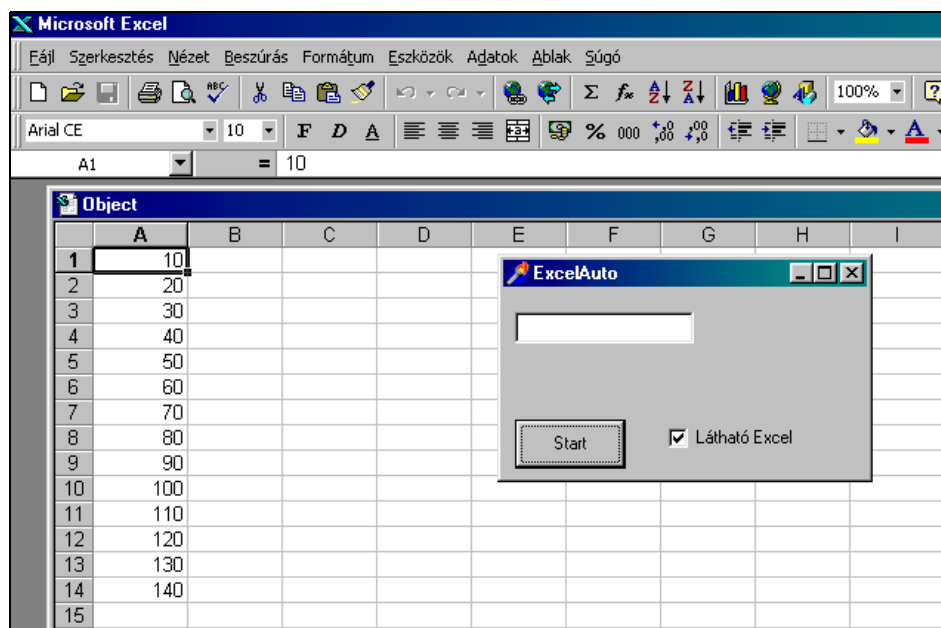
Használjuk a *ComObj* modult, deklaráljunk egy *Variant* típusú változót (*Excel*) az Ole objektumnak és egy konstanst (*MaxN*), ami azt jelzi, hogy hány számot adunk össze.

```
uses comobj;
var
  Excel : Variant;
const
  MaxN=20;
```



A gombnyomásra elindul az *Excel* kitölti az első oszlopot, összeadja a számokat és visszaolvashatjuk az eredményt, majd leállítjuk az *Excel*t.

```
procedure TForm1.Button1Click(Sender: TObject);
var o:integer;
begin
  Edit1.Text:='';
  // Az Excel indítása
  try
    Excel:=0;
    Excel:=CreateOleObject('Excel.Sheet');
    // Szabályozhatjuk, hogy az Excel látható-e
    Excel.Application.Visible:=CheckBox1.Checked;
    // Feltöltjük a cellákat számokkal
    for o:=1 to MaxN do
      Excel.Application.Cells[o,1].Value:=o*10;
    // Az összegző formula
    Excel.Application.Cells[o,1].Formula:='=Sum(A1:A'+inttostr(MaxN)+' )';
    // Az eredmény megjelenítése
    Edit1.Text:=inttostr(Excel.Application.Cells[o,1].value);
    // Az Excel leállítása
    Excel.Application.Quit;
  except
    beep;
  end;
end;
```



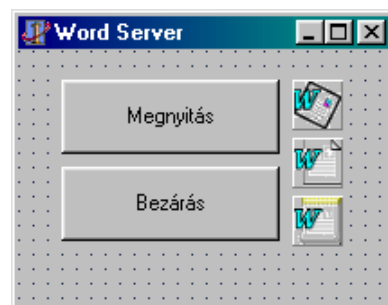


Készítsünk programot, mely adott szöveggel, egy új dokumentummal megnyitja a *Word*-öt! Lehesse bezárni, és újra megnyitni a *Word* alkalmazást! Használjuk a *TOleServer* leszármazottait! (*WordServer*)

A programot ugyanúgy készítjük elő, mint a [WordAuto](#) alkalmazást, azonban most a *Servers* lap három komponensét helyezzük a formra (*WordApplication*, *WordDocument*, *WordFont*). Érdemes megfigyelni, hogy programunkba beépülnek a *Word97* és az *OleServer* modulok is.

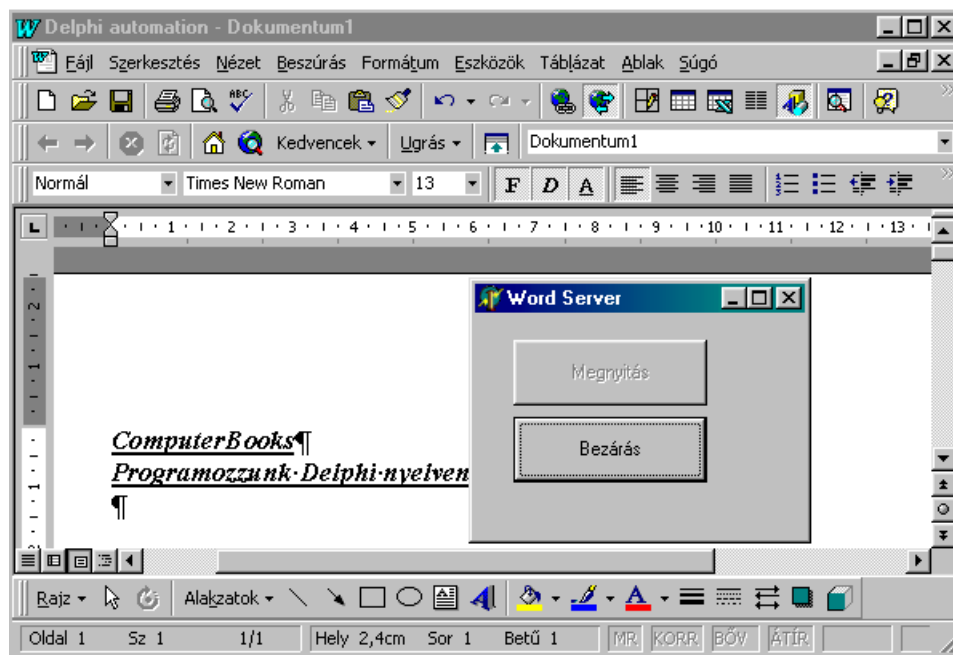
A *Button1* gomb megnyomása indítja a kiszolgálót:

```
procedure TForm1.Button1Click(Sender: TObject);
var Template, NewTemplate, ItemIndex: OleVariant;
begin
  try
    // A Word bekapcsolása
    Template := EmptyParam;
    NewTemplate := True;
    ItemIndex := 1;
    try
      Wordapplication.Connect;
    except
      Abort;
    end;
    Wordapplication.Visible := True; // Látható a Word
    WordApplication.Caption := 'Delphi automation'; // A cím
    //Új dokumentum
    Template := EmptyParam;
    NewTemplate := False;
    WordApplication.Documents.Add(Template, NewTemplate);
    // Hozzárendeljük a dokumentumot a Word komponenshez
    WordDocument.ConnectTo(WordApplication.Documents.Item(ItemIndex));
    // Betűtípus beállítás
    WordFont.ConnectTo(WordDocument.Sentences.Get_Last.Font);
    WordFont.Underline := 1;
    WordFont.Bold := 1;
    WordFont.Italic := 1;
    WordFont.Size := 13;
    WordFont.Name := 'Times New Roman';
    // A szöveg beszúrása
    WordDocument.Range.InsertAfter('ComputerBooks' + #13);
    WordDocument.Range.InsertAfter('Programozunk Delphi nyelven! ' + #13);
    Button1.Enabled:=false;
    Button2.Enabled:=true;
  except
    WordApplication.Disconnect;
  end;
end;
```



A *Button2* pedig bezárja azt.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  SaveChanges,
  OriginalFormat,
  RouteDocument: OleVariant;
begin
  // A kapcsolat megszüntetése és a Word leállítása
  SaveChanges := WdDoNotSaveChanges; // Nincs mentés
  OriginalFormat := UnAssigned;
  RouteDocument := UnAssigned;
  WordFont.Disconnect;
  WordDocument.Disconnect;
  WordApplication.Quit(SaveChanges, OriginalFormat, RouteDocument);
  WordApplication.Disconnect;
  Button1.Enabled:=true;
  Button2.Enabled:=false;
end;
```



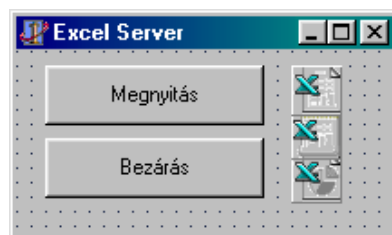




Írjuk Delphi programmal egy Excel táblázatba a Pascal háromszög számait! Használjuk a *TOLeServer* leszármazottait! (*ExcelServer*)

A form hasonlít a [WordServer](#) programéhoz, azonban most az Excel elérésével kapcsolatos komponenseket használjuk.

Az *OleServer* és az *Excel97* modulra való hivatkozás automatikusan bekerül a programba, azonban szükségünk van az *ActiveX* modulra is. Deklaráljunk egy *MaxN* és egy *lcid* egészet, az előző a háromszög méreteit, míg az utóbbi a nyelvet azonosítja.



```
var  
    lcid: integer;  
    MaxN: integer=10;
```

A *Button1* (*Megnyitás*) gomb kapcsolatot teremt a kiszolgálóval. Feltöltjük a Pascal háromszög elemeit.

```
procedure TForm1.Button1Click(Sender: TObject);  
var i,j:integer;  
begin  
    // Az Excel kapcsolása  
    ExcelApplication1.Connect;  
    lcid := GetUserDefaultLCID; // A nyelvi azonosító  
    ExcelApplication1.Visible[lcid]:=True; // Az Excel látható  
    ExcelApplication1.DisplayAlerts[lcid] := False; // Jövőhagyás letiltása  
    // A munkalap és a munkafüzet csatlakoztatása  
    ExcelWorkBook1.ConnectTo(ExcelApplication1.Workbooks.Add(TOLEEnum(xlWBATWorksheet),  
        lcid));  
    ExcelWorkSheet1.ConnectTo(ExcelWorkBook1.Worksheets[1] as _Worksheet);  
    // Cella értékek  
    ExcelWorkSheet1.Range['A1', 'A1'].Value := 'Pascal háromszög';  
    for i:=1 to Maxn do  
        for j:=1 to Maxn-i+1 do  
            begin  
                if (i=1) or (j=1) then  
                    ExcelWorkSheet1.Cells.Item[1+i, 1+j].Value := '1'  
                else  
                    ExcelWorkSheet1.Cells.Item[1+i, 1+j].Value :=  
                        IntToStr(StrToInt(ExcelWorkSheet1.Cells.Item[i, 1+j].Value)+  
                            StrToInt(ExcelWorkSheet1.Cells.Item[1+i, j].Value));  
            end;  
        Button1.Enabled:=false;  
        Button2.Enabled:=true;  
    end;
```

A *Button2* (*Bezárás*) gombbal kilépünk a programból:

```
procedure TForm1.Button2Click(Sender: TObject);  
begin  
    // Lekapcsolódás, leállítás  
    ExcelWorkSheet1.DisConnect;  
    ExcelWorkBook1.DisConnect;  
    ExcelApplication1.Quit;  
    ExcelApplication1.DisConnect;  
    Application.Terminate;  
    Button1.Enabled:=true;  
    Button2.Enabled:=false;  
end;
```

Microsoft Excel

Fájl Szerkesztés Nézet Beszúrás Formátum Eszközök Adatok Ablak Súgó

100%

Arial CE 10 **F** *D* A


% 000 ,00 ,00

A1 = Pascal háromszög

Munka1

	A	B	C	D	E	F	G	H	I	J
1	Pascal háromszög									
2		1	1	1	1	1	1	1	1	1
3		1	2	3	4	5	6	7	8	9
4		1	3	6	10	15	21	28	36	
5		1	4	10	20	35	56	84		
6		1	5	15	35	70	126			
7		1	6	21	56	126				
8		1	7	28	84					
9		1	8	36						
10		1	9							
11		1								
12										
13										
14										

Excel Server

Megnyitás

Bezárás