Visual Basic dla AutoCAD

1. Programowanie

Język programowania to sztuczny język przeznaczony do zapisu algorytmów, w taki sposób, aby mogły one być wykonywane przez komputer. Język programowania charakteryzuje określona składnia (forma) i semantyka (znaczenia).

Efektem programowania jest *kod źródłowy programu,* czyli sformalizowany zapis algorytmu w języku programowania.

Następnie kod źródłowy zostaje przetłumaczony na zapis zrozumiały dla procesora w postaci *kodu maszynowego* (wykonywalnego lub binarnego). Jest to zapis przeznaczony do bezpośredniego wykonania przez procesor i wyrażony w postaci rozumianych przez niego rozkazów oraz ich argumentów.

Do programowania służą środowiska programistyczne w skład, których wchodzą, co najmniej następujące programy: *edytor* tekstowy do pisania kodu źródłowego, edytor graficzny do projektowania graficznego interfejsu użytkownika, *kompilator*, który tłumaczy kod źródłowy na kod maszynowy oraz *debugger* program do śledzenia wykonywania programu w celu poszukiwania błędów.

2. Pojęcia podstawowe

W czasie programowania dane do zadania zapisujemy w programie w postaci *zmiennych,* zaś działania na zmiennych programujemy stosując różnego typu *instrukcje*.

Zmienne to para: nazwa i przyporządkowana jej wartość określonego typu zapisana w pamięci komputera. Nazwa zmiennej powinna być czytelna dla programisty. Przez nazwę programista uzyskuje dostęp do jej wartości zapisanej w pamięci.

Instrukcje jest to najmniejszy samodzielny element języka programowania. Inaczej rozkaz, polecenie dla komputera. Instrukcją może być wykonanie działania arytmetycznego, powzięcie decyzji lub wykonanie pętli obliczeń.

W celu stworzenia zmiennej należy wykonać instrukcję DIM.

```
DIM nazwa_zmiennej AS typ_zmiennej
```

Zmiennej możemy nadać wartość stosując instrukcję przypisania.

```
nazwa_zmiennej = wartość

np. Nazwa_towaru = "drukarka".
```

Do zmiennych możemy również przypisać wyniki obliczania wyrażeń. Wyrażenia powstają przez zastosowanie operatorów, np. arytmetycznych takich jak dodawanie, dzielenie itd.

```
zmienna1 = 2 + 3
```

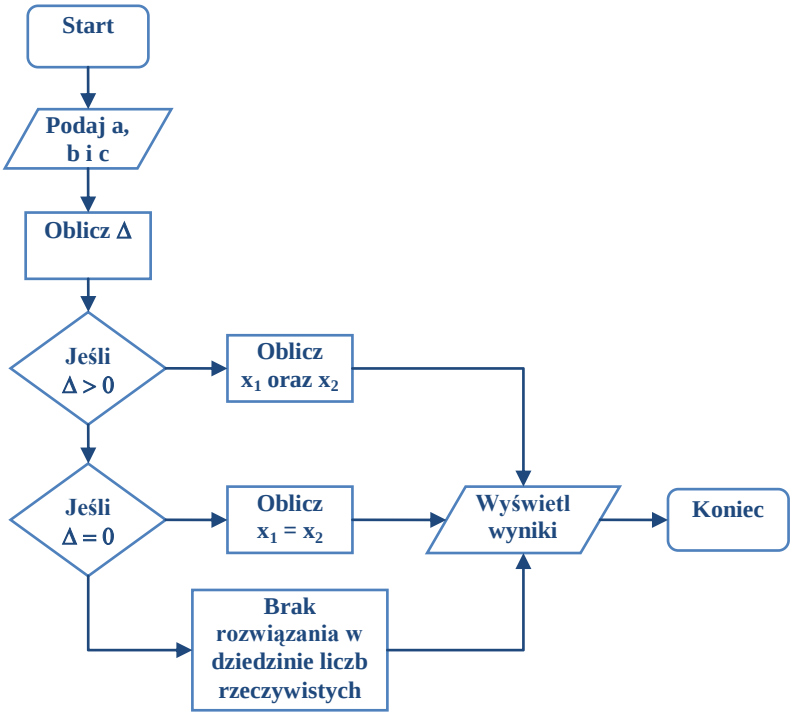
Instrukcje wyboru.

```
If Nazwa_towaru = "Drukarka" Then
    MsgBox "Nazwa towaru to: drukarka."
Else
    MsgBox "Towar to nie jest drukarka"
End If

Select Case Nazwa_towaru
Case "Drukarka"
    MsgBox "Nazwa towaru to: drukarka."
Case "Monitor"
    MsgBox "Nazwa towaru to: monitor."
Case "Klawiatura"
    MsgBox "Nazwa towaru to: klawiatura."
Case Else
    MsgBox "Nieznany towar!"
End Select
```

Instrukcje pętli.

```
For licznik = poczatek to koniec
   MsgBox "Aktualna wartość licznika to: " & licznik
Next licznik.

Do
  odpowiedz = InputBox("Zakończyć pętle (tak/nie)?”)
Loop Until odpowiedz = "tak"
```

Pierwszy program – rozwiązywanie równania kwadratowego.



```
Sub RownanieKwadratowe()

'definicja potrzebnych zmiennych
Dim a As Double
Dim b As Double
Dim c As Double
Dim delta As Double
Dim x1 As String
Dim x2 As String

    'przypisywanie wartosci danych do zmiennych
    a = InputBox("Podaj wartość współczynnika a: ")
    b = InputBox("Podaj wartość współczynnika b: ")
    c = InputBox("Podaj wartość współczynnika c: ")

    'obliczanie wartości delty
    delta = b ^ 2 - 4 * a * c

    ' w zależności od wartości delty obliczamy rozwiązanie
    'jeśli delta jest większa od 0
    If delta > 0 Then
          x1 = (-b - delta ^ 0.5) / (2 * a)
          x2 = (-b + delta ^ 0.5) / (2 * a)
    'w przeciwnym wypadku jeśli delta jest równa 0
    ElseIf delta = 0 Then
          x1 = -b / (2 * a)
          x2 = "x1"
    'w przeciwnym wypadku
    Else
          x1 = "Brak wartości rzeczywistej"
          x2 = " Brak wartości rzeczywistej"
    End If

    'wyświetl wynik
    MsgBox "x1 = " & x1 & ", x2 = " & x2

End Sub
```

3. Programowanie dla aplikacji – model obiektowy aplikacji

Nadrzędnym obiektem jest Aplikacja (*Application*) w ramach, której istnieje zbiór Dokumentów (*Dosuments*).

Każdy Dokument (*Document*) zawiera obszar zwany Przestrzenią Modelu (*ModelSpace*) i Przestrzenią Papieru (*PaperSpace*).

W ramach Przestrzeni Modelu powstają obiekty rysunkowe jak Okrąg (*Circle*), Linia (*Line*) lub Wymiar Kątowy (*DimAngular*).

Każdy obiekt posiada swoje właściwości, metody i zdarzenia. Na przykład obiekt Line pamięta swój początek i koniec odpowiednio we właściwościach *Line.StartPoint*, *Line.EndPoint*. Linia posiada metodę pozwalającą ją przesunąć o nazwie *Line.Move*. Korzystając ze zdarzenia *Line.Modified* programista może zareagować w momencie modyfikacji linii.

4. Tworzenie linii

```
Sub Example_AddLine()

    'deklaracja zmiennych
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double

    'punkt początkowy i końcowy linii
    startPoint(0) = 1#: startPoint(1) = 1#: startPoint(2) = 0#
    endPoint(0) = 5#: endPoint(1) = 5#: endPoint(2) = 0#

    'utowrzenie linii
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)

    ZoomExtents

End Sub
```

5. Tworzenie okręgu

```
Sub Example_AddCircle()

    Dim circleObj As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius As Double

    centerPoint(0) = 0#: centerPoint(1) = 0#: centerPoint(2) = 0#
    radius = 5#

    Set circleObj = ThisDrawing.ModelSpace.AddCircle(centerPoint, radius)
    ZoomAll

End Sub
```

6. Przeglądanie zawartości rysunku

W przykładzie przeglądamy rysunek i wyświetlamy w oknie dialogowym nazwy istniejących obiektów.

```
Sub Example_Count()

    Dim I As Integer

    'wyświetl liczbę obiektów
    MsgBox "Znaleziono " & ThisDrawing.ModelSpace.Count & " obiektów."

    'wyświetl nazwę obiektów
    For I = 0 To ThisDrawing.ModelSpace.Count - 1
      MsgBox "Element numer: " & I & " , to " & _
      ThisDrawing.ModelSpace.Item(I).ObjectName
    Next

End Sub
```

7. Zmiana właściwości obiektów

W przykładzie przeglądamy zawartość rysunku i zmieniamy kolor obiektów. Wszystkim odcinkom kolor zmieniamy na czerwony, a wszystkim okręgom na fioletowy.

```
Sub Change_Color()

    Dim I As Integer

    For I = 0 To ThisDrawing.ModelSpace.Count - 1

        Select Case ThisDrawing.ModelSpace.Item(I).ObjectName
        Case "AcDbLine"
            ThisDrawing.ModelSpace.Item(I).Color = acRed
        Case "AcDbCircle"
            ThisDrawing.ModelSpace.Item(I).Color = acMagenta
        End Select

    Next

End Sub
```

8. Korzystanie ze zbioru wskazań użytkownika

W przykładzie korzystamy ze zbioru wskazań użytkownika. Po wywołaniu funkcji, użytkownik jest proszony o wskazanie elementu. Jeśli jest to okrąg zmieniamy jego promień na 15 jednostek.

```
Sub Edycja_kola()

    'zmienna typu AcadSelectionSet
    Dim ssetObj As AcadSelectionSet

    'usuń wszystkie istniejace zbiory wskazań
    While ThisDrawing.SelectionSets.Count > 0
        ThisDrawing.SelectionSets.Item(0).Delete
    Wend

    'utwórz obiekt zbióru wskazań
    Set ssetObj = ThisDrawing.SelectionSets.Add("KOLO")

    'utwórz zbiór wskazań użytkownika
    ssetObj.SelectOnScreen

    'jesli pierwszy wskazany obiket ma nazwę "AcDbCircle"
    'zmień jego promień
    If ssetObj.Item(0).ObjectName = "AcDbCircle" Then
        ssetObj.Item(0).Radius = 15
    End If

End Sub
```

9. Pobieranie współrzędnych punktu wskazanego przez użytkownika

W przykładzie prosimy użytkownika o wskazanie dwóch punków. Następnie, na podstawie ich współrzędnych tworzymy odcinek.

```
Sub Rysuj_linie()

    Dim firstPnt As Variant
    Dim secondPnt As Variant
```

```
    firstPnt = ThisDrawing.Utility.GetPoint(, "Wskaż pierwszy punkt: ")
    secondPnt = ThisDrawing.Utility.GetPoint(, "Wskaż drugi punkt: ")

    Dim lineObj As AcadLine
    Set lineObj = ThisDrawing.ModelSpace.AddLine(firstPnt, secondPnt)

    ZoomAll

End Sub
```

W poniższym przykładzie, w punkcie wskazanym przez użytkownika stworzymy tekst "Dzień dobry"

```
Sub Pisz_tekst()

    Dim firstPnt As Variant
    Dim textObj As AcadText

    firstPnt = ThisDrawing.Utility.GetPoint(, "Enter a point: ")

    Dim textString As String
    Dim height As Double

    'Tworzymy tekst w przestrzeni modelu
    textString = "Dzien dobry"
    height = 0.5

    Set textObj = ThisDrawing.ModelSpace.AddText(textString, firstPnt, height)

    ZoomAll

End Sub
```

10. Gra *Kółko i krzyżyk*

Poniższa funkcja rysuje planszę do gry w miejscu wskazanym przez użytkownika.

```
Sub Szachownica()

    'zmienna dla lewego, dolnego punktu planszy
    Dim basePnt As Variant

    'zmienne dla końców odcinków tworzących plansze
    'są to tablice zawierające X (indeks 0), Y (indeks 1) i Z (indeks 2)
    Dim Pnt1(0 To 2) As Double
    Dim Pnt2(0 To 2) As Double
    Dim Pnt3(0 To 2) As Double
    Dim Pnt4(0 To 2) As Double
    Dim Pnt5(0 To 2) As Double
    Dim Pnt6(0 To 2) As Double
    Dim Pnt7(0 To 2) As Double
    Dim Pnt8(0 To 2) As Double
    Dim Pnt9(0 To 2) As Double

    'Definiujemy lewy, dolny narożnik planszy
    basePnt = ThisDrawing.Utility.GetPoint(, "Enter a point: ")

    'Definiujemy współrzędne wszystkich punktów
```

```
    Pnt1(0) = basePnt(0) + 10: Pnt1(1) = basePnt(1): Pnt1(2) = 0
    Pnt2(0) = basePnt(0) + 10: Pnt2(1) = basePnt(1) + 30: Pnt2(2) = 0
    Pnt3(0) = basePnt(0) + 20: Pnt3(1) = basePnt(1): Pnt3(2) = 0
    Pnt4(0) = basePnt(0) + 20: Pnt4(1) = basePnt(1) + 30: Pnt4(2) = 0
    Pnt5(0) = basePnt(0): Pnt5(1) = basePnt(1) + 10: Pnt5(2) = 0
    Pnt6(0) = basePnt(0) + 30: Pnt6(1) = basePnt(1) + 10: Pnt6(2) = 0
    Pnt7(0) = basePnt(0): Pnt7(1) = basePnt(1) + 20: Pnt7(2) = 0
    Pnt8(0) = basePnt(0) + 30: Pnt8(1) = basePnt(1) + 20: Pnt8(2) = 0
    Pnt9(0) = basePnt(0) + 30: Pnt9(1) = basePnt(1) + 30: Pnt9(2) = 0


    'Tworzymy linie
    Dim lineObj As AcadLine
    Set lineObj = ThisDrawing.ModelSpace.AddLine(Pnt1, Pnt2)
    Set lineObj = ThisDrawing.ModelSpace.AddLine(Pnt3, Pnt4)
    Set lineObj = ThisDrawing.ModelSpace.AddLine(Pnt5, Pnt6)
    Set lineObj = ThisDrawing.ModelSpace.AddLine(Pnt7, Pnt8)

    'Powiększamy obraz do rozmiarów szachownicy
    ZoomWindow Pnt1, Pnt9

End sub
```

Funkcje rysującą planszę wykorzystamy we właściwej procedurze gry w kółko i krzyżyk. Należy ja najpierw zmodyfikować o możliwość przekazania danych o punkcie w którym rysujemy planszę. Zmienną basePnt wpisujemy jako parametr wywołania funkcji, a zmienna lokalną basePnt usuwamy.

```
Sub Szachownica(basePnt As Variant)

    Dim basePnt As Variant
```

Procedura gry w kółko i krzyżyk będzie wyglądała następująco

```
Sub Kolko_krzyzyk()

    Dim basePnt As Variant
    Dim Znak As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double
    Dim markerPnt(0 To 2) As Double

    'rysujemy szachownice za pomocą wcześniej przygotowanej procedury
    'zmienną basePoint wykorzystamy później do lokalizacji wskazań na planszy
    Szachownica basePnt

    'zaczyna kółko
    Znak = "0"
    'licznik ruchów
    licznik = 1
    'wysokość znaków
    height = 5

    Do
      'prosimy o pierwszy ruch
      nextPnt = ThisDrawing.Utility.GetPoint(, "Enter a point: ")

      'oblicz współrzędne środka wskazanego pola
      markerPnt(0) = ((nextPnt(0) - basePnt(0)) \ 10) * 10 + 2.5 + basePnt(0)
      markerPnt(1) = ((nextPnt(1) - basePnt(1)) \ 10) * 10 + 2.5 + basePnt(1)
```

```
      markerPnt(2) = 0

      'jesli wskazanie jest w obszarze planszy
      If nextPnt(0) > basePnt(0) And nextPnt(0) < basePnt(0) + 30 And _
         nextPnt(1) > basePnt(1) And nextPnt(1) < basePnt(1) + 30 Then

         'zdecyduj jaki postawić znak
         If Znak = "0" Then
            Znak = "X"
         Else
            Znak = "0"
         End If
         'narysuj znak
         ThisDrawing.ModelSpace.AddText Znak, markerPnt, height
         'zwiększ licznik ruchów
         licznik = licznik + 1
      End If
    'realizuj pętlę tak długo aż użytkownik wykona poprawnie 9 ruchów
    Loop Until licznik > 9

   End Sub
```

11. Współpraca z innymi aplikacjami

Poniższa procedura odczytuje z arkusza Excela rodzaj obiektu oraz jego współrzędne i na tej podstawie rysuje obiekty w przestrzeni modelu.

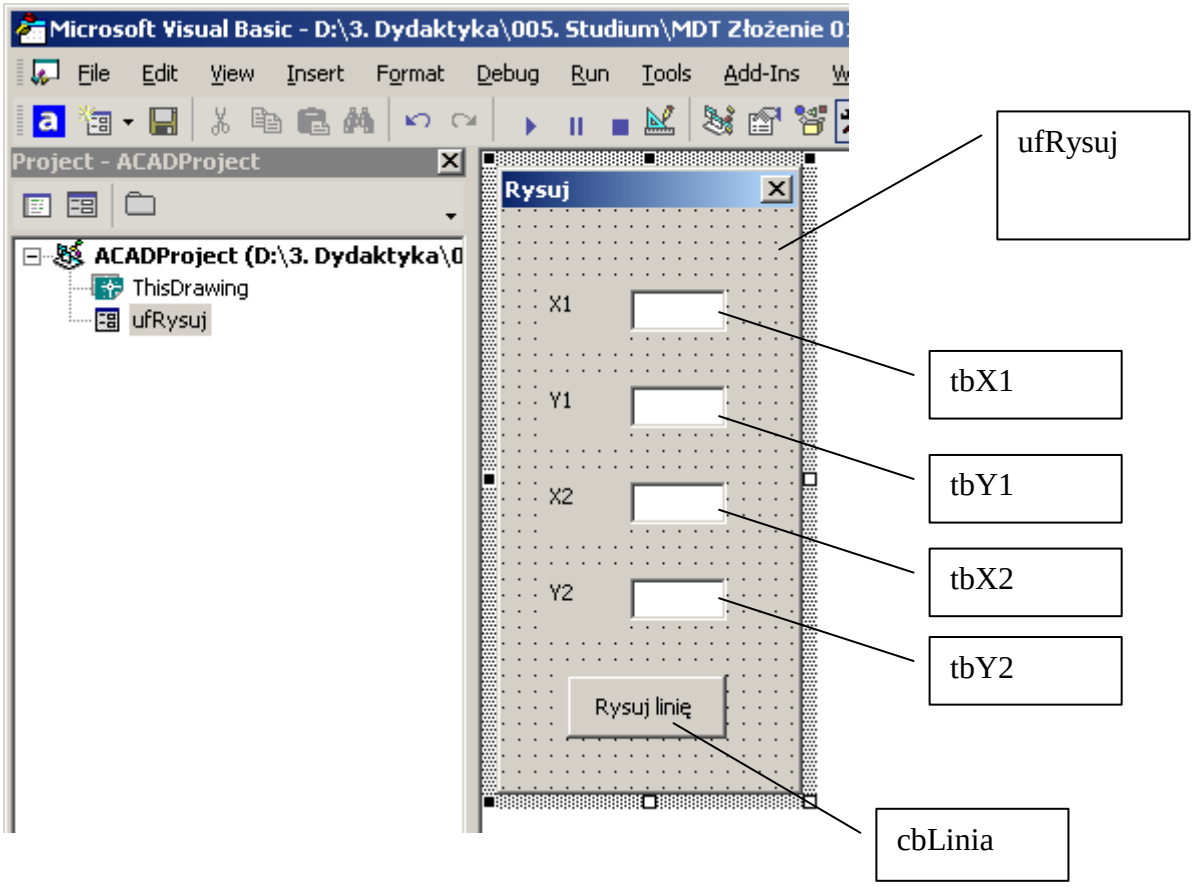
Przygotuj plik Excela. W pierwszej kolumnie wpisz nazwy tworzonych obiektów „Linia” lub „Okrag”. W kolejnych kolumnach wpisz, dla linii współrzędne końców odcinka, dla okręgu współrzędne środka i promień.

```
Sub Excel()

    Dim Pnt1(0 To 2) As Double
    Dim Pnt2(0 To 2) As Double
    Dim lineObj As AcadLine

    Dim Promien As Double
    Dim circleObj As AcadCircle

    'Przygotuj plik Excela
    Set ExcelWorksheet = GetObject("C:\...\Arkusz.xls")

    I = 1
    Do
      Select Case ExcelWorksheet.activesheet.cells(I, 1)
            'Jeśli w pierwszej komórce jest „Linia”,
            'wczytaj współrzędne końców i narysuj odcinek
            Case "Linia"
                Pnt1(0) = ExcelWorksheet.activesheet.cells(I, 2)
                Pnt1(1) = ExcelWorksheet.activesheet.cells(I, 3)
                Pnt2(0) = ExcelWorksheet.activesheet.cells(I, 4)
                Pnt2(1) = ExcelWorksheet.activesheet.cells(I, 5)
                Set lineObj = ThisDrawing.ModelSpace.AddLine(Pnt1, Pnt2)

            'Jeśli w pierwszej komórce jest „Okrag”,
            'wczytaj współrzędne środka oraz promienia i narysuj okrąg
            Case "Okrag"
                Pnt1(0) = ExcelWorksheet.activesheet.cells(I, 2)
```

```
                Pnt1(1) = ExcelWorksheet.activesheet.cells(I, 3)
                Promien = ExcelWorksheet.activesheet.cells(I, 4)
                Set circleObj = ThisDrawing.ModelSpace.AddCircle(Pnt1, Promien)

        End Select

        I = I + 1

'Czytaj dopóki nie napotkasz pustej komórki
Loop Until ExcelWorksheet.activesheet.cells(I, 1) = ""

End Sub
```

12. Interfejs użytkownika dla polecenia rysowania linii

Utwórz okno dialogowe i nazwij je ufRysuj. W oknie wstaw elementy interfejsu i nazwij je tak jak na poniższym rysunku.



Utwórz procedurę, która będzie uruchomiona po naciśnięciu przycisku cbLinia. W tym celu dwukrotnie kliknij na przycisk „Rysuj linię”.

```
Private Sub cbLinia_Click()

    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double

    ' zdefiniuj początek i koniec odcinka pobierając dane z okna dialogowego
```

```
    startPoint(0) = ufRysuj.tbX1
    startPoint(1) = ufRysuj.tbY1
    startPoint(2) = 0#
    endPoint(0) = ufRysuj.tbX2
    endPoint(1) = ufRysuj.tbY2
    endPoint(2) = 0#

    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    ZoomAll

End Sub
```

W folderze ThisDrawing definiujemy funkcję wywołującą okno rysowania odcinka.

```
Sub RysowanieElementów()
    ufRysuj.Show
End Sub
```

13. Edycja okręgu

W niniejszym przykładzie, za pomocą okna dialogowego, użytkownik będzie miał możliwość modyfikacji promienia okręgu.

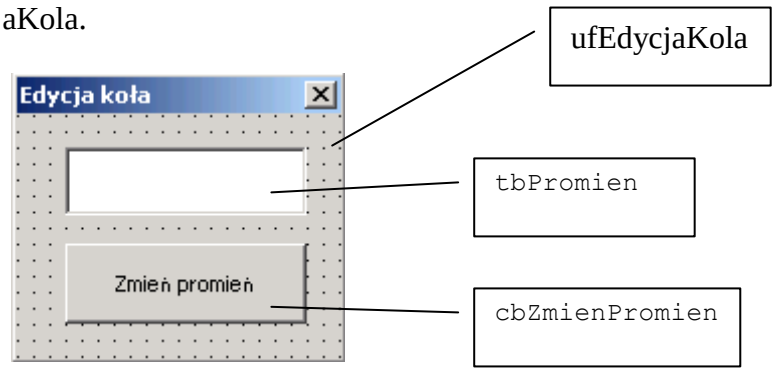
Chcemy aby polecenie edycji okręgu było dostępne z menu przyciskowego. Do utworzonego przycisku przypisujemy makro:

```
^C^C-vbarun ThisDrawing.EdycjaKola
```

W folderze ThisDrawing definiujemy funkcję wywołującą okno.

```
Sub EdycjaKola()
    ufEdycjaKola.Show
End Sub
```

Tworzymy okno dialogowe ufEdycjaKola.



W folderze ufEdycjaKola definiujemy następujące zmienne i funkcje:

Zmienną zbioru wskazań:

```
Public ssetObj As AcadSelectionSet
```

W czasie uruchomienia okna prosimy użytkownika o wskazanie obiektu. Dlatego w procedurze UserForm_Initialize wpisujemy:

```
Private Sub UserForm_Initialize()

    'Usuwamy wszystkie istniejace zbiory wskazań
    If ThisDrawing.SelectionSets.Count > 0 Then
        ThisDrawing.SelectionSets.Item(0).Delete
    End If

    'Tworzymy nowy zbiór wskazań
    Set ssetObj = ThisDrawing.SelectionSets.Add("KOLO")

    ssetObj.SelectOnScreen

    'Wartość promienia wskazanego okręgu wpisujemy do pola dialogowego
    If ssetObj.Item(0).ObjectName = "AcDbCircle" Then
        ufEdycjaKola.tbPromien = ssetObj.Item(0).Radius
    End If

End Sub
```

Po naciśnięciu przycisku "Zmień promień" chcemy zmienić promień na wpisany przez użytkownika w pole edycyjne. Odpowiednio wypełniamy metodę Click przycisku cbZmienPromien.

```
Private Sub cbZmienPromien_Click()

    If ssetObj.Item(0).ObjectName = "AcDbCircle" Then
        ssetObj.Item(0).Radius = ufEdycjaKola.tbPromien
        ssetObj.Item(0).Update
    Else
        MsgBox("To nie jest okrąg!" & vbCr & „Spróbuj ponownie.")
    End If

End Sub
```