

# VISUAL BASIC 6 KURS

autorzy:  
Krystian Ł.  
Kamil R.

wznowienie kursu:  
Krystian Ł.

wersja z: 05.11.2005

strona główna kursu:  
<http://visual.basic.kaem.forall.pl/>

# SPIS TREŚCI

1. Informacje podstawowe .....	5
2. Środowisko VB .....	5
3. Zmienne.....	6
4. Stałe .....	8
5. Właściwości formy.....	8
6. MsgBox.....	11
7. Przyciski .....	12
8. Właściwości obiektów.....	14
9. Label.....	16
10. Pole tekstowe.....	16
11. CheckBox .....	17
12. OptionButton .....	17
13. Suwaki .....	18
14. ComboBox .....	18
15. Image .....	19
16. Shape & Line.....	19
17. Operacje na stringach .....	20
18. Funkcje matematyczne .....	22
19. Instrukcje warunkowe .....	23
20. Pętle .....	27
21. Procedury .....	29
22. Funkcje .....	30
23. Tablice .....	30
24. Moduły .....	32
25. Forma .....	33
26. ListBox .....	34
27. DriveListBox, DirListBox, FileListBox.....	36
28. Timer .....	37
29. ListView .....	37
30. TreeView .....	43
31. Dostęp do plików .....	47
32. Operacje na plikach.....	48
33. Operacje na katalogach .....	49
34. Atrybuty plików i folderów .....	50
35. Spis plików i katalogów .....	51
36. Obsługa błędów .....	53
37. Schowek Windows.....	54
38. Klawisze.....	55
39. Shell.....	57
40. Rejestr.....	58
41. Menu.....	59
42. Pasek statusu .....	61
43. Menu graficzne.....	62
44. Czas .....	64

45. Indeksowanie obiektów .....	66
46. Przeciąganie obiektów.....	68
47. Forma – rysowanie .....	69
48. Forma – autoredraw .....	70
49. Forma – pisanie .....	71
50. O API.....	72
51. Przeciąganie obiektów.....	73
52. Własny kształt formularza.....	75
53. GetPixel i SetPixel .....	76
54. BitBlt i StretchBlt.....	77
55. Operacje na polu tekstowym .....	78
56. If .....	79
57. Split i Join.....	79
58. Type.....	80
59. With.....	83
60. Zdarzenia .....	84
61. Input Box .....	86
62. Menu Podręczne .....	87
63. Option Explicit .....	89
64. Praktyka – Zmienne .....	89
65. Praktyka – Suwaki.....	91
66. Praktyka - Używanie kolorów .....	92
67. Graficzny pasek postępu .....	93
68. Obsługa błędów + http .....	94
69. WinSock.....	98
70. Połączenie IRC .....	99
71. Połączenie <u>FTP</u> .....	101
72. FTP, komendy i listowanie.....	107
73. FTP, transfer danych .....	114
74. FTP, listowanie^3.....	116
75. Połączenie HTTP.....	117
76. Algorytm: NWD.....	119
77. Algorytm: Sortowanie Bąbelkowe .....	121
78. Przeszukiwanie Binarne .....	123
79. Drukowanie – Podstawy.....	126
80. Drukowanie - Obróbka tekstu .....	128
81. Drukowanie – Grafika.....	135



# 1. Informacje podstawowe

<http://visual.basic.kaem.forall.pl/>

Witaj, jeżeli zacząłeś to czytać to na pewno jesteś zainteresowany(a) programowaniem w środowisku graficznym (RAD). Zanim wybierzesz język Visual Basic musisz poznać parę jego wad i zalet.

**Zalety :**

- + Szybkie tworzenie aplikacji
- + Intuicyjne poruszanie się w menu
- + Łatwe wykorzystanie bibliotek DirectX (Można uzyskać nawet 120 FPS)
- + Proste wykorzystywanie Windows API
- + Możliwość tworzenia własnych kontrolerek
- + Łatwy dostęp do rejestru, plików ini i sieci

**Wady :**

- Jest wolniejszy od języków C o około 8 % (dane ze strony Pmasters)
- Nie pozwala tworzyć aplikacji dosowych
- Nie ma bezpośredniego połączenia z Assemblerem

## 2. Środowisko VB

<http://visual.basic.kaem.forall.pl/>

Gdy pierwszy raz włączamy program wyskakuje nam okienko z wyborem aplikacji, na razie wybieramy 'Standard EXE' i klikamy otwórz. Teraz widzimy już główne okno programu (powinno być takie jak na obrazku (oprócz kolorów oczywiście))

**Kolor Czerwony (Tolbox)** - Jest to lista obiektów, aby umieścić jakiś obiekt na formie wystarczy kliknąć w wybrany obiekt i postawić go w wybranym miejscu (kliknięcie + przeciągnięcie). Obiekty możemy dodawać, wystarczy wybrać opcje z menu : Project > Components... lub wcisnąć CTR+T, pokazuje się nam okienko z różnymi obiektami, jeśli jakiś chcemy to wystarczy go odhaczyć i zastosować zmiany.

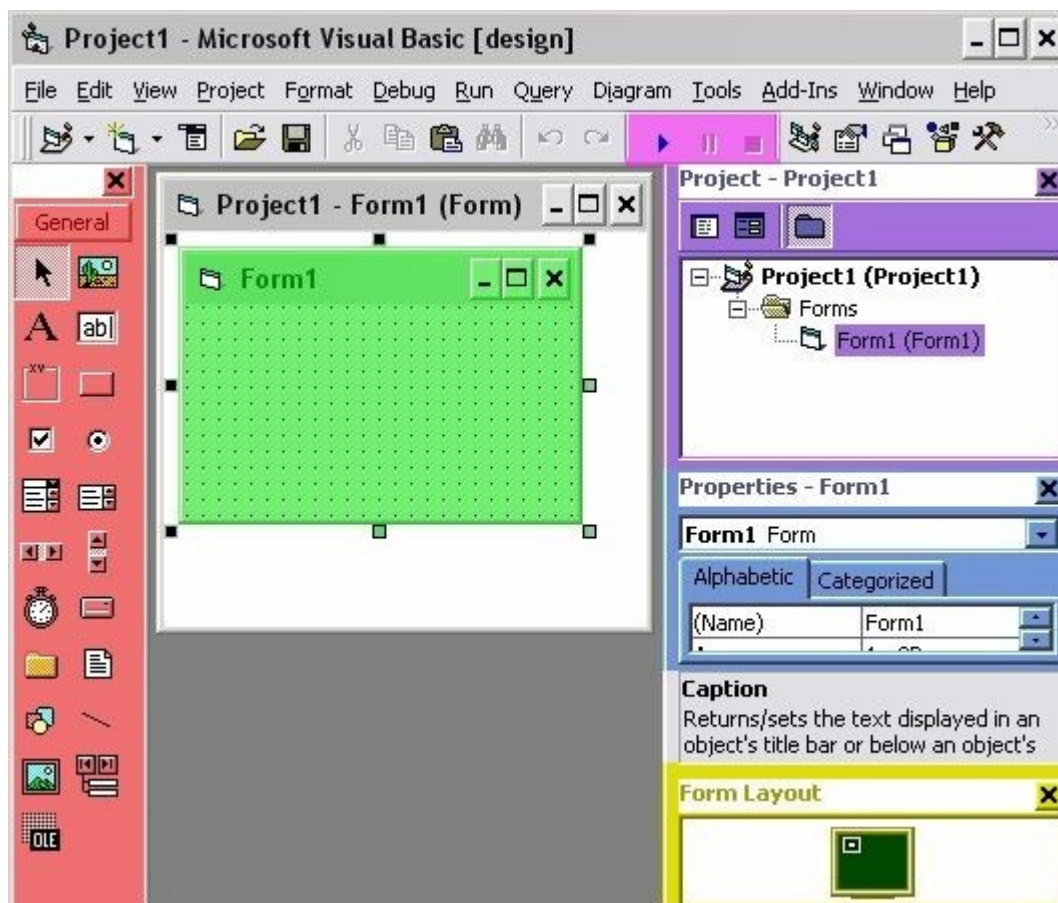
**Kolor Zielony (Form)** - W tym okienku widzimy wygląd aplikacji, a także kod źródłowy, możesz tutaj umieszczać różne obiekty z Tolbox-u

**Kolor Różowy (Start)** - Tym przyciskiem testujemy aplikacje, możemy to także robić wciskając F5

**Kolor Fioletowy (Project Explorer)** - Tu widzimy wszystkie formy, moduły itp. Znajdują się tutaj także 2 przyciski, jeden włącza okienko kodu, a drugi okienko wyglądu formy

**Kolor Niebieski (Properties)** - Tutaj widzimy różne właściwości obiektów, możemy je łatwo edytować i zmieniać im nazwy

**Kolor Żółty (Form Layout)** - Okienko w którym możemy ustalić miejsce otwarcia okienka na ekranie



### 3. Zmienne

<http://visual.basic.kaem.forall.pl/>

#### 1. Zmienne-co to takiego?

Zmienna w fachowym określeniu jest wydzieloną częścią pamięci, o rozmiarze określonym przez jej typ. Oznacza to, że zmienna jest to taki pojemnik, w którym przechowujemy dane. Dane te można odczytywać, zapisywać i zamieniać. Jeżeli zmiennej przyporządkujemy jakąś wartość, to w miejscach jej wywołania zostanie zastąpiona swoją wartością. Trochę to pogmatwane, ale zaraz wszystko się wyjaśni. Zmienne wywołujemy poprzez podanie ich nazwy. Oto przykład pozwalający to wszystko zrozumieć:

`wiek = 4` ' przyporządkowujemy zmiennej wiek wartość 4  
`wiek = wiek + 2` ' przyporządkowujemy zmiennej wiek wartość o 2 większą od jej poprzedniej - czyli 6

Z matematycznego punktu widzenia ten drugi zapis jest niepoprawny, znak = nie jest znakiem równości tylko znakiem przyporządkowania. Zmienna wiek przyjmuje wartość zmiennej wiek + 2.

#### 2. Typy zmiennych

Sama nazwa zmiennej podczas deklaracji nie wystarczy, ponieważ program nie będzie

wiedział ile pamięci dla niej zarezerwować. Visual Basic pozwala zdefiniować typ zmiennej na jeden z dostępnych poniżej. Są to standardowe typy zapewnione przez Visual Basic:

Typ	Liczba bajtów	Zastosowanie
Boolean	2	True lub false
Byte	1	Liczby całkowite od 0 do 255
Integer	2	Liczby całkowite od -32768 do 32767
Long	4	Liczby całkowite +/-2000000000
Currency	8	Liczby o określonej liczbie cyfr po przecinku
Single	4	Liczby dziesiętne do 7 cyfr znaczących
Double	8	Liczby dziesiętne do 14 cyfr znaczących
Date	8	Liczby dziesiętne reprezentujące czas i godzinę
String	1bajt/znak	Wartości tekstowe
Variant	16	Wartości dowolnego typu

### 3.Zasięg zmiennej

Zmienną możemy zadeklarować jako prywatną, czyli widoczną tylko w bloku (wybranej formie/module), w którym ją zadeklarowano , zmienne publiczną, dostępną z każdego miejsca w programie lub zmienne tymczasowe. Aby zadeklarować zmienną prywatną na samym początku musimy napisać Private, jeśli jest to zmienna publiczna to Public, dla zmiennej tymczasowej będzie to Dim

Znając ta wszystkie zasady możemy przystąpić do deklaracji. Ma to postać:

Private liczba As Integer  
Public znaki As String  
Dim tymczasowa As String

Zmienne Prywatne i Publiczne deklarujemy zaraz przy samej górze kodu, zmienne tymczasowe deklarujemy w procedurach/funkcjach/zdarzeniach, gdyż podczas opuszczenia zdarzenia zmienne tego ulegają wyładowaniu z pamięci.

Private Zmienna As Integer

Private Sub Form\_Load()

Dim Zmienna2 As Integer  
Zmienna2 = 8

Zmienna = 6 + Zmienna2

End Sub

Zmienna2 zadeklarowana w zdarzeniu Form\_Load zniknie z pamięci po wykonaniu zdarzenia, natomiast Zmienna zadeklarowana przez Private "nazewnątrz" procedur będzie stale dostępna dla programu.

### 4.Prefixy zmiennych.

Istnieje jeszcze coś takiego, jak prefixy zmiennych. Można je stosować dodając bezpośrednio do nazw zmiennych zamiast As typ. Oto prefixy:

Typ	Prefix
Integer	%
Long	&
Single	!
Double	#
Currency	@
String	\$

Byte, Boolean, Date, Variant nie posiadają prefixów

Wynikałoby z tego, że zamiast zapisu:

`Dim kolor_oczu As String`

możemy napisać:

`Dim kolor_oczu$`

Nie polecam jednak tego sposobu, ze względu na to, że wielu początkującym może sprawić trudności, w odgadywaniu typów, a poza tym kod zostaje wzbogacony o tzw. 'krzaczkę'.

Mimo tego jest to efektywne i nauczanie się takiego deklarowania zmiennych może zaoszczędzić znaczne ilości czasu.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (2)

## 4. Stałe

<http://visual.basic.kaem.forall.pl/>

Stałe, powiedzmy są to specyficzne odmiany zmiennych, których wartość możemy określić tylko raz i to przy deklaracji. W zamian jednak nie musimy deklarować ich typu. Do deklaracji stałych używamy słowa Const. Dla stałych także istnieje możliwość opisu zakresu widoczności. Np.:

`Private Const WIEK = 15`

`Public Const IMIĘ = 'Marcin'`

Zwykło się pisać zmienne małymi, a stałe dużymi literami. Poprawia to czytelność kodu.

## 5. Właściwości formy

<http://visual.basic.kaem.forall.pl/>

### Appearance

Appearance	Określa, czy forma ma być trójwymiarowa, czy płaska	0-Flat płaska 1- trójwymiarowa
BackColor	Kolor tła formy	



BorderStyle	Styl formy (obramowanie)	0- None- brak obramowania, paska tytułu, przycisków sterujących. Stosowane w oknach komunikatów.
Caption	Tekst na pasku tytułowym formy	
FillColor	Kolor wypełnienia formy	
FillStyle	Styl wypełniania figur na formie	0-Solid- pełny 1-Transparent- przezroczysty 2-Horizontal Line- linie poziome 3-Vertical Line- linie poziome 4-Upward Diagonal- linie skośne w prawą stronę 5-Downward Diagonal- linie skośne w lewą stronę 6-Cross- linie pionowe i poziome skrzyżowane (połączenie 2 i 3) 7-Diagonal Cross- linie skośne z prawej i lewej strony skrzyżowane (połączenie 4 i 5)
FontTransparent	Określa widoczność tła pod napisami i obrazkami na formie	True- niewidoczne False- widoczne
ForeColor	Kolor tła pod tekstem i obrazkami	
Palette	Określa obrazek z kolorami jakie mają być wyświetlane	
Picture	Obrazek wyświetlany w oknie formy jako tło	

### Behavior

AutoRedraw	Odświeżanie okna	True- okno jest odświeżane False-okno nie jest odświeżane
DrawMode	Tryb rysowania po formie	
DrawStyle	Styl rysowania po formie	
DrawWidth	Wielkość pędzla do rysowania po formie	
Enabled	Określa czy okno można uaktywnić klikając na nie	True- można False- nie można
HasDC	Czy forma posiada uchwyt	
OLEDropMode	Tryb przeciągania metodą OLE	
PaletteMode	Rodzaj palety	
RightToLeft	Wyświetlanie liter wspak	True- wyświetlaj wspak False- wyświetlaj normalnie
Visible	Określa widoczność formy	True- forma jest widoczna False- forma nie jest widoczna

**Misc**

Name	Określa nazwę kontrolki	
ControlBox	Wyświetlanie przycisków minimalizuj, maksymalizuj, zamknij	True- są widoczne False- nie są dostępne
HelpContextID	Numer identyfikacyjny obiektu przypisany do pliku pomocy	
Icon	Ikona wyświetlana na uchwycie okna	
KeyPreview	Określa, czy forma ma przechwytywać wszystkie znaki z klawiatury	True- przechwytuj False- nie przechwytuj
MaxButton	Określa, czy przycisk maksymalizacji ma być dostępny	
MDIChild	Mówi kompilatorowi, czy forma jest wyświetlana jako okno potomne	
MinButton	Określa dostępność przycisku minimalizuj	
MouseIcon	Określa plik z kształtem kursora wyświetlanego po najechaniu na okno formy (widoczne, gdy MousePointer jest ustawione na 99)	
MousePointer	Zmienia kształt kursora na jeden ze standardowych Windows lub zdefiniowany przez MouseIcon	0-Default- domyślny 1-Arrow- strzałka 2-Cross- krzyżyk 3-taki jak przy wprowadzaniu tekstu w Wordzie. 4-Icon- ??? 5-Size- jak przy przenoszeniu obiektów 6... 99 - zdefiniowany przez MouseIcon
NegotiateMenus	Wyświetlanie menu	
ShowInTaskbar	Dostępność na pasku zadań Windows	
Tag	Zbiornik przechowujący dane	
WindowState	Określa wielkość okna po uruchomieniu	0-okno normalnej wielkości 1-okno zminimalizowane 2-okno zmaksymalizowane

**Position**

Height	Wysokość okna	
Left	Odległość okna od lewej strony ekranu	
Moveable	Mówi, czy można przemieścić okno, przeciągając za pasek tytułowy	True- można False- nie można
StartPosition	Określa położenie okna, po uruchomieniu programu	
Top	Określa odległość okna od góry ekranu	
Width	Szerokość okna	

## Scale

ScaleHeight	Liczba jednostek wysokości okna (jednostkę ustawia ScaleMode)
ScaleLeft	Liczba jednostek odległości od lewej krawędzi okna
ScaleMode	Ustala aktualną jednostkę
ScaleTop	Liczba jednostek od góry ekranu
ScaleWidth	Szerokość okna wyrażana jednostkami ustalonymi za pomocą ScaleMode

## 6. MsgBox

<http://visual.basic.kaem.forall.pl/>

Msgbox jest funkcją wyświetlającą komunikat, jej zastosowanie wygląda tak

```
a = MsgBox("Komunikat",przyciski+ikona,"Tytuł")
```

Wyświetli to nam okienko o nazwie Tytuł i o treści Komunikat  
Spośród przycisków i ikon możemy wybrać :

VbCritical - ikona krytyczna

VbQuestion - znak zapytania

VbExclamation - żółty ostrzegawczy trójkąt

VbInformation - ikona informacji

VbSystemModal - mała ikona systemowa

VbOKOnly - tylko przycisk ok

VbOKCancel - przyciski ok. i anuluj

VbAbortRetryIgnore - przyciski przerwij, ponów i ignoruj

VbYesNoCancel - przyciski tak nie anuluj

VbYesNo = przyciski tak nie

VbRetryCancel przyciski ponów, anuluj

Wymagany jest tylko 1 parametr, czyli treść komunikatu, pozostałych parametrów nie musimy podawać.

```
a = MsgBox("Witaj")
```

Jeżeli nie chcemy później sprawdzić jaką opcję z komunikatu wybrał użytkownik lub nie musimy tego robić nie trzeba podawać zmiennej która przechowuje wynik, można to zrobić w ten sposób:

```
Msgbox "Witaj"
```

Przykładowy komunikat wyglądałby tak :

```
a = MsgBox("Witaj użytkownikowi",VbOKOnly,"Powitanie")
```

Można też łączyć przyciski i ikony znakiem +

```
a = MsgBox("Podoba ci się program",VbYesNo + VbInformation,"Pytanie")
```

Zmienna a przechowuje nam kliknięty przycisk, oto ich numery :

1 - ok

- 2 - anuluj
- 3 - przerwij
- 4 - ponów
- 5 - ignoruj
- 6 - tak
- 7 - nie

Możemy więc sprawdzić jak użytkownik odpowiedział na nasze pytanie. Musisz jednak dojść do lekcji If.. Then

Różnica między wyświetlaniem funkcji z rezultatem oraz bez jest następująca:

- gdy chcemy uzyskać z funkcji jakąś wartość piszemy:

Wartosc = Funkcja("Parametr1","Paramater2",...)

- gdy nie potrzebujemy uzyskać wartości:

Funkcja "Parametr1","Paramter2",....

Parametry oddzielamy przecinkami, jeżeli chcemy pominąć jakiś paramter zostawiamy poprostu puste miejsce i piszemy sam przecinek:

Funkcja "Paramter1", , "Paramter3"

Funkcja , , , "Paramter4"

Niektóre parametry są obowiązkowe, niektórych nie musimy podawać, wszystko zależy od funkcji.

Visual basic informuje o parametrach które nie są wymagane, pokazuje je w nawiasie kwadratowym:

Msgbox

MsgBox(**Prompt**, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult

Na obrazku mozna zobaczyć że dla funkcji MsgBox tylko 1 parametr jest wymagany, reszta nie jest obowiązkowa

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 7. Przyciski

<http://visual.basic.kaem.forall.pl/>

Aby umieścić przycisk (CommandButton) na formie wystarczy go narysować w wybranym miejscu. Następnie jeśli chcemy ustalić kod który ma się pokazać po kliknięciu przycisku klikamy w niego dwukrotnie (w przycisk na formie oczywiście). Pojawi nam się :

Private Sub Command1\_Click()

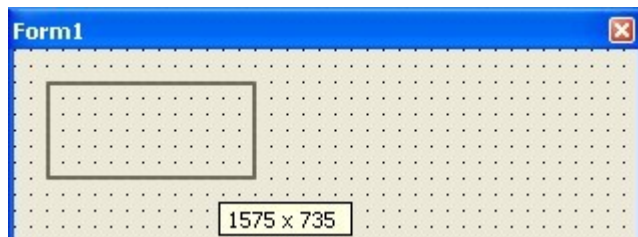
End Sub

Umieszczanie elementów na formie polega na:

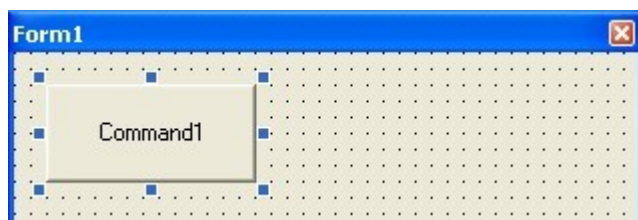
1. wybraniu elementu (kliknięcie go), jeżeli nie jesteśmy pewni czy wybraliśmy dobry element zatrzymajmy na nim na chwile kursor a pojawi nam się jego nazwa



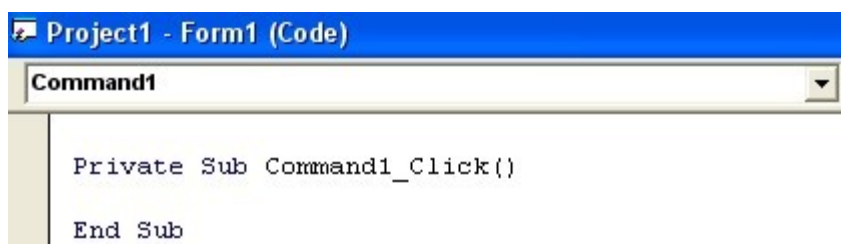
2. następnie zaznaczamy obszar na formie aby określić rozmiar elementu (niektóre elementy nie mogą mieć dowolnych rozmiarów), robimy to zwyczajnie klikając myszką i przeciągając ją tworząc wybrany kształt



3. puszczamy przycisk myszki i na naszej formie pojawia się obiekt o standardowej nazwie (zazwyczaj nazwa elementu + numer)



4. aby przejść do standardowego zdarzenia dla obiektu należy go dwukrotnie kliknąć (czasem niektóre dodatkowe właściwości są dostępne przez kliknięcie prawym myszy na obiekt i wybranie Properties)



Możemy w środek wpisać dowolne polecenie lub wiele poleceń, na przykład MsgBox

```
Private Sub Command1_Click()  
a = MsgBox("Klikłeś mnie", "Informacja")  
End Sub
```

Po kliknięciu pojawi się komunikat:



Najważniejszymi właściwościami dla przycisku są :

- Caption - tekst na naszym przycisku
- Enabled - ustala czy obiekt jest aktywny
- Picture - obrazek na przycisku
- ToolTipText - napis pojawiający się po najechaniu
- Font - czcionka napisu na przycisku

Spróbujcie poeksperymentować z innymi właściwościami wiele ich powtarza się z innymi obiektami więc bezsensu było by ich powtarzanie.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

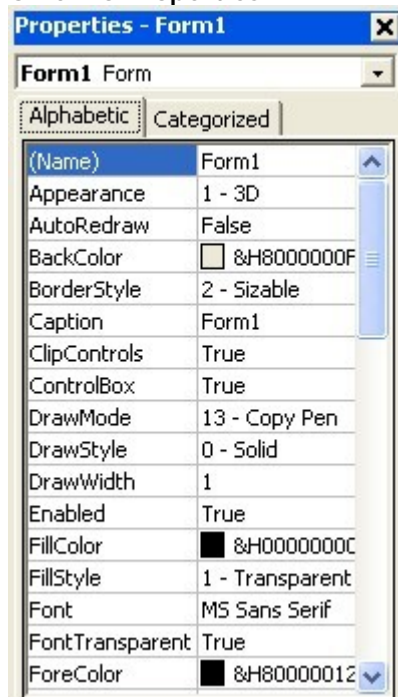
Można znaleźć przykłady do tej lekcji (1)

## 8. Właściwości obiektów

<http://visual.basic.kaem.forall.pl/>

Każdy obiekt ma swoje różne właściwości. Możemy je zmieniać w okienku Properties.

Okienko Properties:



Główną zaletą właściwości (nie wszystkich) jest to że można je zmieniać w trakcie działania programu.

Najważniejszą właściwością jest Name czyli nazwa naszego obiektu. Umieścimy na formie przycisk i zmienimy mu właściwość name na Przycisk. Właściwości obiektów możemy także zmieniać w trakcie działania programu robimy to stosując przypisanie:

**Nazwaobektu.Właściwość = co**

Przykładowo naszemu przyciskowi o Name = Przycisk zmienimy napis tak :

**Przycisk.Caption = "Nowy napis na przycisku"**

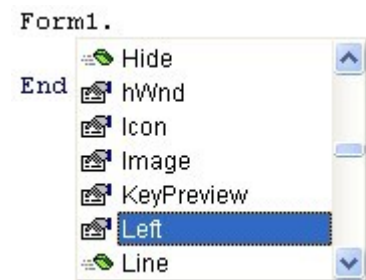
Dla przykładu umieścimy na formie dwa przyciski, niech ich nazwy będą niezmiennione (Command1 i Command2). Drugiemu przyciskowi ustawmy właściwość Enabled = False i napiszmy taki oto kod :

```
Private Sub Command1_Click()  
Command2.Enabled = True  
Command1.Enabled = False  
End Sub  
Private Sub Command2_Click()  
Command1.Enabled = True  
Command2.Enabled = False  
End Sub
```

Po kliknięciu w przycisk 1 robi się on nieaktywny, a przycisk 2 robi się aktywny. Możemy zmieniać właściwości dla każdego obiektu. Możemy też zmieniać właściwości obiektów znajdujących się na innej formie :

**Forma.Obiekt.Właściwość = NowaWłaściwość**

Właściwości można wybrać z rozwijanej listy (jeśli przypadkiem zapomnimy dokładnej nazwy właściwości)



Lista pojawia się w momencie wpisania kropki.

Standardowe właściwości :

- Left - położenie obiektu z lewej strony
- Top - położenie obiektu od góry
- Width - szerokość obiektu
- Height - wysokość obiektu
- Enabled - jeśli True to obiekt jest aktywny
- Aligment - wyrównanie
- Visible - jeśli True to obiekt jest widoczny

Visual Basic określa położenie obiektów i ich rozmiary w Twipsach, a nie Pikselach. Możemy to zmienić we właściwościach formy ScaleMode. Możemy też nauczyć się przeliczać piksele na twpisy i odwrotnie.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 9. Label

<http://visual.basic.kaem.forall.pl/>

Label jest kontrolką wyświetlającą tekst.

Ważniejsze właściwości :

Caption - wyświetlany tekst

BackStyle - tło (0 - przezroczyste, 1 - normalne)

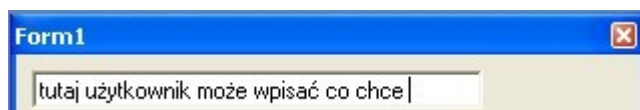
Font - rodzaj i wielkość czcionki

Alignment - pozycja tekstu (0 - do lewej, 1 - do prawej, 2 - tekst wyśrodkowany)

## 10. Pole tekstowe

<http://visual.basic.kaem.forall.pl/>

Pole tekstowe (TextBox) to pole które pozwala wpisywać użytkownikowi tekst.



Ważniejsze właściwości :

Text - przechowuje wpisany tekst

MultiLine - jeżeli true to jest to pole wielowierszowe

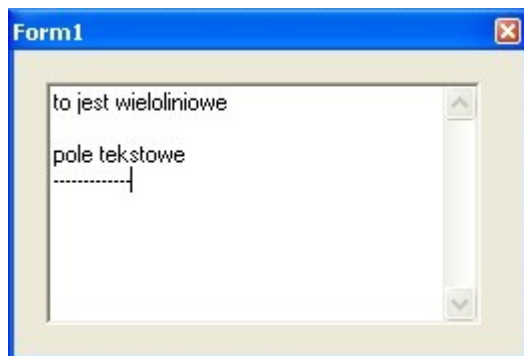
PasswordChar - możemy tu określić znak maskowania. nie widać wtedy wpisanego tekstu tylko nasz znak

Przykład : Na formie umieszczamy przycisk (Command1) i pole tekstowe (Text1). Obok pola ustawiamy label o treści wpisz swoje imię. Teraz klikamy dwa razy na przycisk i piszemy :

```
Private Sub Command1_Click()  
    MsgBox "witaj " & Text1.Text  
End Sub
```

Gdy ustawimy MultiLine = True przydało by też dodać suwaki, służy do tego parametr Scrollbars, ustawiamy go na wartość 1/2/3 (suwak poziomy/suwak pionowy/oba suwaki).





Aby samemu przejść do nowej linii w polu tekstowym należy użyć kombinacji znaków Chr(13) i Chr(10)

```
Text1.Text = "Pierwsza linia" & Chr(13) & chr(10) & "Druga linia"
```

Kombinacja tych znaków jest przypisana do stałej vbCrLf, używanie tego jest szybsze:

```
Text1.Text = "Pierwsza linia" & vbCrLf & "Druga linia" & vbCrLf & "Trzecia linia"
```

## 11. CheckBox

<http://visual.basic.kaem.forall.pl/>

Checkbox jest polem do zaznaczania (zachaczania), najczęściej używany do wyboru opcji przez użytkownika

**Ważniejsze właściwości :**

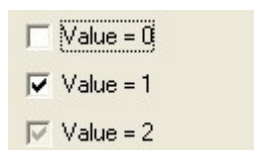
Caption - tekst przy kwadracie

Value - zaznaczenie:

0 - brak zaznaczenia

1 - zaznaczony

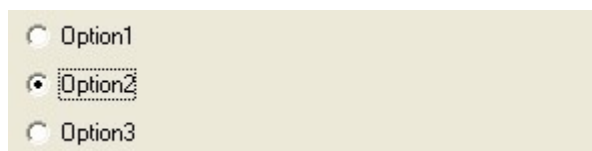
2 - zaznaczony i niekatywny



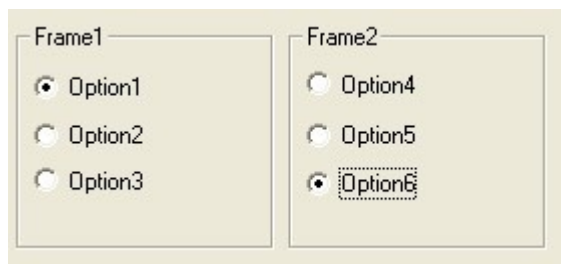
## 12. OptionButton

<http://visual.basic.kaem.forall.pl/>

Option jest elementem prawie identycznym do ChceckBox, różnicą jest tylko to, że umieszczając wiele obiektów OptionButton tylko jeden może być zaznaczony.



Jeżeli chcemy na jednej formie ustawić więcej niż jeden "słupek" `OptionButton`-ów tak aby dało się wybrać jeden element z pierwszego słupka i jeden z drugiego, musimy ułożyć `OptionButtons` w osobnych warstwach (np. w `PictureBoxie` lub `Frame`)



## 13. Suwaki

<http://visual.basic.kaem.forall.pl/>

### Ważniejsze właściwości :

Min - najmniejsza wartość suwaka

Max - największa wartość

SmallChange - o ile miejsc ma być przesunięty suwak po kliknięciu w strzałkę

LargeChange - o ile miejsc ma być przesunięty suwak przy dużej zmianie

Value - przechowuje aktualną wartość suwaka pomiędzy Min a Max

Przykład :

Ustaw na formie suwak (przypisz mu name na Suwak) i label. Ustaw suwakowi :

Min = 0

Max = 100

SmallChange = 1

LargeChange = 5

Kliknij dwukrotnie na suwak i wpisz :

`Label1.Caption = Suwak.Value`

Dzięki temu Label1 będzie na bieżąco wyświetlał wartość suwaka.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 14. ComboBox

<http://visual.basic.kaem.forall.pl/>

`ComboBox` to rozwijana lista, właściwie pole tekstowe do którego możemy i pisać i wybrać coś z listy.

### Ważniejsze właściwości :

Text - tekst przy comboboxie

List - lista elementów do wyboru, możemy je dopisywać we właściwościach lub używając

polecenia `AddItem (Combo1.AddItem "tekst")`, więcej o `AddItem` poczytasz przy `ListBox Style` - możemy ustawić je na 0 (standardowa), 1 czyli sam tekst i 2 czyli element musi być wybrany

**Elementy :**

Ilość elementów jest przechowywana w parametrze `ListCount`, chcąc przypisać do jakiejś zmiennej ilość elementów wystarczy napisać:

```
Ile = Combo1.ListCount
```

Wybrany element wywołujemy poprzez polecenie `Combo1.List(NumerElementu)`, elementy liczone są od 0:

```
Element2 = Combo1.List(1)
```

Jeżeli chcemy się dowiedzieć jaki element został wybrany przez użytkownika wystarczy użyć zmiennej przechowującej numer zaznaczonego elementu, czyli `ListIndex`. Chcąc wywołać zaznaczony element należy więc napisać:

```
Zaznaczony = Combo1.List(Combo1.ListIndex)
```

## 15. Image

<http://visual.basic.kaem.forall.pl/>

`Image` to zwyczajne pole obrazka, klikając na właściwość `Picture` możemy wybrać dowolny obrazek. Obrazki możemy też zmieniać w trakcie działania programu :

```
Image1.Picture = LoadPicture("plik.obrazka.bmp")
```

Możemy także ładować do obrazka `Image1` obrazek z innego pola `Obrazka`

```
Image1.Picture = Image2.Picture
```

Obrazki zapisujemy przez procedurę `'SavePicture Obrazek,Ścieżka'`, niestety format zapisywania taką metodą to `.bmp`

```
SavePicture Image1.Picture, "C:/obrazek.bmp"
```

## 16. Shape & Line

<http://visual.basic.kaem.forall.pl/>

`Shape` pozwala nam rysować figury geometryczne, z właściwości `Shape` możemy wybrać rodzaj figury, są to :

- 0 - prostokąt
- 1 - kwadrat
- 2 - elipsa
- 3 - koło
- 4 - prostokąt o zaokrąglonych bokach
- 5 - kwadrat o zaokrąglonych bokach

Line rysuje prostą linię

## 17. Operacje na stringach

<http://visual.basic.kaem.forall.pl/>

### LCase

Funkcja zamienia wszystkie duże litery w tekście na małe

```
tekst = "TeKst z dużymi lItErAmi"
```

```
tekst = LCase(tekst)
```

tekst po obróbce będzie wyglądał : "tekst z dużymi literami"

### UCase

Ta funkcja zaś zamienia małe litery na duże

```
tekst = "Jakiś napis"
```

```
tekst = UCase(tekst)
```

Teraz nasz tekst to "JAKIŚ NAPIS"

### Left

Funkcja wycina wybraną ilość liter od lewej strony wybranego napisu

```
tekst = "Testowy wpis"
```

```
tekst = Left(tekst,4)
```

Da nam : "Test"

### Right

Funkcja działa tak samo jak powyższa tyle że wycina znaki od prawej strony

```
tekst = "Nasz napis"
```

```
tekst = Right(tekst,3)
```

Da nam "pis"

### Mid

To już bardzo zaawansowana funkcja wycinająca tekst, struktura : Mid(tekst,start,ile).

Funkcja wycina tekst ze zmiennej tekst, zaczynając od miejsca start tyle znaków ile zadeklarujemy w ile.

```
tekst = "Styl nad style"
```

```
tekst = Mid(tekst,3,6)
```

Znak startowy to y i wycina sześć znaków czyli zwróci nam "yl nad"

### Chr

Zastanawialiście się jak wyświetlić na komunikacie MsgBox tekst w postaci :

```
"Witaj ! W moim nowym programie"
```

Do tego posłuży nam funkcja Chr, wyświetla ona znaki z tablicy ASCII (jest to tablica zawierająca znaki klawiatury, każdy z tych znaków ma swój numer).

```
a = MsgBox(chr(34) & "Witaj ! W moim nowym programie" & chr(34))
```

chr(34) zwróci nam znak " którego normalnie nie moglibyśmy wyświetlić

### Asc

Asc zwraca nam numer klawisza

```
a = MsgBox(Asc("."))
```

Powinien się wyświetlić komunikat 46, gdyż jest to numer kropki z tablicy ascii

### Trim

Usuwa boczne spacje z napisu :

```
napis = Trim(" Tekst ")
```

Otrzymamy "Tekst"

Gdy chcemy uciąć spacje tylko z lewej lub prawej strony używamy odpowiednio LTrim lub RTrim

### Replace

Kolejna ważna funkcja, użycie jej wygląda następująco : tekst =

Replace(tekst,szukane,naco), zamienia to wszystkie szukane słowa na słowo naco,

przykład :

tekst = "To jest testowy tekst"

tekst = Replace(tekst,"jest","miał być")

Teraz nasz tekst wygląda tak : "To miał być testowy tekst". Zamieniliśmy napis "jest" na "miał być".

### Len

Zwraca nam ilość znaków w tekście

tekst = "Visual"

ile = Len(tekst)

Zwróci nam liczbę 6 bo napis Visual ma 6 znaków

### InStr

Dzięki temu możemy dowiedzieć się gdzie w danym tekście leży dana litera/ciąg znaków, wygląda to tak : InStr(start,tekst,litera). Start to miejsce od której litery chcemy szukać.

Przykład :

tekst = "koc"

pozycja = InStr(1,tekst,"c")

Zwróci nam pozycję litery czyli 3. Gdy chcemy znaleźć kolejne c to musieli byśmy napisać InStr(4,tekst,"c"), tyle że w wyrażeniu nie ma drugiego c i funkcja zwróci nam 0.

### Val

Val zamienia tekst na liczbę (tylko do rozmiaru Long)

Dim Tekst As String

Dim Liczba As Long

Tekst = "-2390938"

Liczba = Val(Tekst)

### Like

Jest to dosyć trudne polecenie, sprawdza ono poprawność znaków, zaczniemy od 1 litery:

litera = "C"

pop = litera Like "C"

C to C więc wartość true

pop = litera Like "[A-Z]"

Sprawdza czy C jest w przedziale od A do Z, jest czyli też true

pop = litera Like "[0-9]"

Sprawdza czy C jest cyfrą, nie jest czyli pop przyjmie wartość false

pop = litera Like "#"

Sprawdza czy C jest znakiem, czyli True

Przedziały sprawdzania mogą być dowolne np. [d-k], [4-9]. Teraz zajmijmy się wyrazami:

wyraz = "Test"

pop = wyraz Like "T\*"

Da nam True, ponieważ pierwszy znak to T i występuje po nim dowolna ilość znaków (\*)

pop = wyraz Like "T"

Teraz otrzymamy False, gdyż aby było True musiała by być tylko 1 litera

pop = wyraz Like "T\*t"

True, gdyż pierwsze K, potem dowolna ilość znaków i na końcu t

pop = wyraz Like "K#S"

False, # różni się od \* tym, że jest dla jednego znaku, tak więc wyraz musiałby być trzyliterowy

pop = wyraz Like "[A-Z][a-z][a-z][a-z]"

True, pierwsza litera duża od A do Z, potem mała, mała, mała

pop = wyraz Like "[m-t]t"

Także True, bo na początku x znaków, potem sprawdza czy s jest w przedziale od m do t i na końcu t.

Dzięki tej funkcji możemy sprawdzić poprawność wypełnienia pól przez użytkownika, sprawdzając np. czy użytkownik podał prawidłowo kod pocztowy powinniśmy użyć  
czyok = Text1.Text Like "[0-9][0-9]-[0-9][0-9][0-9]"

## 18. Funkcje matematyczne

<http://visual.basic.kaem.forall.pl/>

Jest kilka funkcji oferowanych przez Visual Basic do operacji na liczbach. Każda z nich zwraca jakąś wartość i jej wywołanie ma postać:

Zmienna = funkcja(parametr)

Oto one:

Int(liczba)	Zwraca część całkowitą liczby dziesiętnej
Abs(liczba)	Zwraca wartość bezwzględna z liczby
Sgn(liczba)	Daje w wyniku 1 dla liczby dodatniej, -1 dla ujemnej, 0 dla 0
Fix(liczba)	Zwraca część całkowitą
Sqr(liczba)	Zwraca pierwiastek kwadratowy
Log(liczba)	Zwraca logarytm naturalny z liczby
Exp(liczba)	Zwraca liczbę e podniesioną do potęgi liczba
Sin(kąt)	Zwraca sinus kąta w radianach
Cos(kąt)	Zwraca cosinus kąta w radianach
Rnd	Zwraca liczbę losową z przedziału 0 do 1
Round(liczba,mpp)	Zaokrągla naszą liczbę, mpp to ilość miejsc po przecinku

Przykładowe wywołanie funkcji :

a = Sin(90)

Od teraz nasza zmienna a to wartość sinusa z kąta 90 stopni.

Wypadało by napisać o Rnd coś więcej. Rnd zwraca losową liczbę z przedziału od 0 do 1 w postaci ułamkowej do dalekiego miejsca po przecinku (np. 0.7612347912379). Chcąc więc wylosować liczbę w przedziale od 0 do 100 możemy napisać :

`liczba = Rnd * 100`

Aby efekt losowania był większy możemy użyć funkcji losowania i zaokrąglania.

**Randomize**

`liczba = Round(Rnd * 100)`

Gdybyśmy nie użyli Randomize to program losowałby zawsze te same liczby z kolei. Przykładowo po pierwszym włączeniu : 40,15,78,12,26, po drugim : 40,15,78,12,26 itd.. Randomize po prostu temu zapobiega.

Aby wylosować liczbę z wybranego przedziału liczbowego należy poprostu zastosować wzór:

`liczba = Round(Rnd * (Maks - Min) + Min)`

Wylosowanie liczby z przedziału od 70 do 100:

`liczba = Round(Rnd * (100-70) + 70)`

Albo prościej wykonując od razu odejmowanie i opuszczając nawias:

`liczba = Round(Rnd * 30 + 70)`

## 19. Instrukcje warunkowe

<http://visual.basic.kaem.forall.pl/>

Instrukcje warunkowe służą do sterowania pracą programu. Jest kilka typów warunków. Oto pierwszy z nich:

### 1. Znaki porówniania:

- = równość
- > większe niż
- < mniejsze niż
- >= większe lub równe
- <= mniejsze lub równe
- <> różne od

### 2. If.....Then.....Else.....EndIf

Instrukcja ta służy do sprawdzania warunków (jednego lub więcej) i na podstawie zwróconych wartości wykonywania odpowiednich bloków programów. Oto schemat tej instrukcji:

**If (warunek) Then**  
**Instrukcje**  
**EndIf**

Po słowie If następuje warunek ujęty w nawias, a po nim słowo Then rozpoczynające blok instrukcji, w przypadku poprawności warunku. Później są instrukcje zakończone EndIf, nakazującym programowi wyjście z warunku. Inny, bardziej rozbudowany przykład to:

```
If (warunek) Then
Instrukcje1
Else
Instrukcje2
EndIf
```

Od poprzedniego przykładu różni się tylko wystąpieniem słowa Else. Instrukcje po nim są wykonywane, jeśli warunek będzie nieprawdziwy. Np.:

```
Wiek=14
If (Wiek > 15) Then
b = 15
Else
b = 14
EndIf
```

Warunek nie musi się składać tylko z jednego członu. Można je łączyć operatorami logicznymi, np:

```
If ((Wiek > 15) AND (Imie = "Jan")) Then
b = 2
Else
b = 1
End If
```

Inną możliwością oferowaną przez instrukcję If jest jej zagnieżdżenie. Ma to postać:

```
If (wiek > 15) Then
b = 15
Else
  If (imie <> "George") Then
    c = 1
  Else
    c = 0
  EndIf
b = 14
EndIf
```

Przykład : Ustawmy na formie pole tekstowe (Text1, wartość Text = 0) i jeden przycisk (Command1), możemy też dać label z napisem wpisz w pole swój wiek

Po kliknięciu w przycisk piszemy

```
If (Text1.Text < 20) Then
Msgbox("Jesteś jeszcze młody")
End If
If (Text1.Text > 19) And (Text1.Text < 50) Then
Msgbox("Jesteś w średnim wieku")
End If
If (Text1.Text > 49) Then
Msgbox("Młodzieńcze lata masz już za sobą")
End If
```

Można też to uprościć (skrócić)

```
If (Text1.Text < 20) Then MsgBox("Jesteś jeszcze młody")
If (Text1.Text > 19) And (Text1.Text < 50) Then MsgBox("Jesteś w średnim wieku")
```



**If (Text1.Text > 49) Then MsgBox("Młodzieńcze lata masz już za sobą")**

Napisanie tego powyższymi dwoma sposobami nie jest zbyt poprawne, program najpierw sprawdzi jeden warunek i jeżeli się zgadza to wyświetli komunikat, potem drugi warunek i trzeci. To aż 3 sprawdzenia za każdym razem. Prościej można to napisać na opisanym trochę dalej Case lub stosując ElseIf (Inaczej Jeżeli)

```
If Text1.Text < 20 Then ' 1  
MsgBox ("Jesteś jeszcze młody") ' 2  
ElseIf Text1.Text > 19 And Text1.Text < 50 Then ' 3  
MsgBox ("jesteś w średnim wieku") ' 4  
Else ' 5  
MsgBox ("Młodzieńcze lata masz już za sobą") ' 6  
End If
```

1. jeżeli Text1.Text jest mniejszy od 20 to
2. wyświetl komunikat
3. inaczej jeżeli Text1.Text jest większy niż 19 i mniejszy niż 50 to
4. wyświetl komunikat
5. jeszcze w innym przypadku
6. wyświetl komunikat

ElseIf można oczywiście używać obojętną ilość razy

Teraz gdy zgodzi się jakiś z warunków program nie sprawdza już kolejnych tylko opuszcza całe If

### 3. Operatory logiczne

Głównymi operatorami których się używa są:

**AND** - I  
**OR** - lub  
**NOT** - Nieprawda (zaprzeczenie)  
**XOR** - (tylko jeden warunek ma się zgadzać)

**If A = 6 AND B = 7 Then**  
- warunek jest spełniony jeżeli A = 6 i B = 7

**If A = 6 OR B = 7 Then**  
- warunek jest spełniony jeżeli A = 6 lub B = 7 lub A = 6 i B = 7

**If A = 6 XOR B = 7 Then**  
- warunek jest spełniony jeżeli A = 6 i B nie jest równe 7 lub gdy B = 7 i A nie jest równe 6

**If NOT A = 6 Then**  
- warunek jest spełniony gdy A nie jest równe 6 (NOT dodajemy zawsze na początku warunków)

Łącząc operatory logiczne można otrzymać ciekawe zestawienia, np:

**If NOT A = 6 XOR B = 7 Then**  
- ten warunek jest spełniony tylko wtedy gdy A = 6 i B = 7 lub wtedy gdy A <> 6 i B <> 7, gdy jeden warunek się zgadza (albo A, albo B) to warunek nie jest spełniony.

#### 4. Case

Instrukcja Case służy do szybkiego sprawdzania jednej wartości z wieloma dostępnymi. Oczywiście możliwe jest stosowanie drabinki If...Else....If..... ale niepotrzebnie to komplikuje program. Postać Case jest następująca:

```
Select Case wyrażenie
Case test1
Instrukcje1
Case test2
Instrukcje2
.....
Case Else
Instrukcje
End Select
```

W każdym Case następuje sprawdzenie warunku. Jeśli wynikiem jest True, to następuje wykonanie instrukcji pod tym wyrażeniem. Jeśli warunek zwraca False (jest nieprawdziwy) to program przechodzi do sprawdzenia następnego bez wykonywania instrukcji. Jeśli wszystkie sprawdzenia będą nieprawdziwe, to zostanie wykonana instrukcja po Case Else. Człon ten jest jednak nie obowiązkowy i nie musi być stosowany. I oczywiście mały przykładzik:

```
Select Case wiek
Case 7
a = 1
Exit
Case sqr(11)
a = 2
Exit
Case 2 To 10
a = 3
Exit
Case "A" To "Z"
a = 4
Exit
Case "15"
a = 5
Exit
Case wiek > 6
a = 6
Exit
Case Else
a = 7
End Select
```

Wyrażeniem testowym może być:

liczba lub wyrażenie liczbowe (np.: 7, sqr(10))  
ciągami znaków lub wyrażeniem (np.: "Tak", nazwa)  
przedziałem wartości (np.: 2 To 6, "A" To "Z")  
porównaniem (np.: Is > 6, Is < "M")

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

# 20. Pętle

<http://visual.basic.kaem.forall.pl/>

## 1. Pętle

Pętle pozwalają nam uniknąć niepotrzebnych powtórzeń kodu, przez co program staje się mniejszy, bardziej zwięzły i często szybszy. Jest kilka rodzajów pętli. Aby zrozumieć ich działanie, należy wcześniej zapoznać się z warunkami. Przedstawię je po kolei wszystkie i krótko opiszę:

## 2. Pętla For

Pierwszą pętlą, którą poznamy jest for. Jest ona najprostsza w działaniu i wymaga jednej zmiennej. Jej składnia jest następująca:

For zmienna = pocz To kon Step krok  
Instrukcje  
Next zmienna

Po słowie For następuje przyporządkowanie początkowej wartości zmiennej wcześniej zadeklarowanej. Pętla od tej właśnie wartości rozpocznie działanie. Kon oznacza wartość zmiennej przy której pętla zakończy powtarzanie. Step określa o ile zmienna będzie zmieniać swoją wartość po każdym wywołaniu zawartych w niej instrukcji. Jeśli nie ma tego parametru standardowo przyjmowana jest wartość 1. Wyrażenie Next zmienna zmienia wartość zmiennej o tą podaną w Step i pokazuje programowi gdzie kończą się instrukcje do powtórzenia. Mały przykład:

Dim koniec As Byte  
Dim i As Byte  
For i = 0 To 9 Step 1  
MsgBox i  
Next i

Spowoduje to 10-krotne wyświetlenie okienka z wartością i (od 0 do 9)

## 3. Do

Pętli Do jest pięć. Są jej różne odmiany, różniące się na pierwszy rzut oka niewiele od siebie. Ich możliwości też są podobne i znajomość nawet jednej z nich może zastąpić pozostałe pięć, ale warto się zaprzyjaźnić z nimi wszystkimi.

a) Pierwsza z nich ma postać:

Do While (warunek)  
Instrukcje  
Loop

Instrukcje są powtarzane tak długo, jak warunek jest prawdziwy. Jeśli nie jest spełniony na początku, to pętla nie jest wykonywana ani razu.

b) Druga to:

Do  
Instrukcje  
Loop While (warunek)

Pętla jest wykonywana do momentu, gdy warunek jest fałszywy. Zawsze jest wykonywana przynajmniej jeden raz.

c) Trzecia:

Do Until (warunek)  
Instrukcje  
Loop

Pętla jest wykonywana do czasu spełnienia warunku. Jeśli warunek jest prawdziwy przy pierwszym wywołaniu, to pętla nie jest wykonywana ani razu.

d) Czwarta:

Do  
Instrukcje  
Loop Until

Instrukcje są wywoływane do momentu spełnienia warunku. Zawsze jest wykonywana przynajmniej jeden raz.

e) Piąta, ostatnia:

Do  
Instrukcje  
Loop

Jest wykonywana do czasu natrafienia na instrukcję Exit lub naciśnięcia klawiszy CTRL+BREAK.

Przykład (ja zawsze stosuje tylko For i Do While (tą właśnie opisze)) :

```
a = 0
Do While (a < 20)
a = a + 1
Msgbox("Pętla działa " & a & " raz")
Loop
```

- 1 linijka - a przyjmuje wartość 0
- 2 linijka - powtarzaj zawsze gdy a jest mniejsze od 20
- 3 linijka - zwiększ a o 1
- 4 linijka - wyświetl komunikat o numerze powtórzenia
- 5 linijka - koniec deklaracji pętli

Pętla za każdym razem zwiększa wartość a o 1 gdy a jest mniejsze od 20 to pętla się powtarza, gdy natomiast a już nie jest mniejsze od 20 to pętla kończy swoją działalność.

Funkcja licząca silnie za pomocą pętli:

```
Private Function Silnia(Ile As Byte)
Dim Start As Byte
Dim Wartosc As Double
Wartosc = 1
Start = 1
Do While Start < Ile
Start = Start + 1
```

```
Wartosc = Wartosc * Start
Loop
Silnia = Wartosc
End Function
```

Tak więc wywołanie funkcji `Wartosc = Silnia(8)` zwróci nam wartość 40320 (bo  $1*2*3*4*5*6*7*8 = 40320$ , każde mnożenie oprócz pierwszego dokonuje się za każdym razem gdy pętla się powtarza)).

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 21. Procedury

<http://visual.basic.kaem.forall.pl/>

Procedura jest to obszar kodu programu odpowiedzialny za wykonanie określonych zadań. Standardowo w VB procedury wyglądają tak :

```
Private Sub Command1_Click()
```

```
End Sub
```

`Command1_Click` to nazwa procedury, potem jest miejsce na jej polecenia i `End Sub` jako jej zakończenie. Procedury mogą mieć też swoje zmienne :

```
Private Sub Command1_Click(Index As Integer)
```

```
End Sub
```

Istnieje także możliwość tworzenia własnych procedur z własnymi parametrami :

```
Private Sub Komunikat(Tekst As String)
MsgBox Tekst
End Sub
```

Teraz naszą procedure wywołujemy tak :

```
Komunikat "Testowy komunikat"
```

Nasze procedury mogą być publiczne (dla całego projektu programu) i prywatne (tylko dla tego form co je deklarujemy)

```
Private Sub Komunikat(Tekst As String)
MsgBox Tekst
End Sub
```

Prywatna procedura tylko dla naszego form

```
Public Sub Komunikat(Tekst As String)
MsgBox Tekst
End Sub
```

## 22. Funkcje

<http://visual.basic.kaem.forall.pl/>

Funkcje tym różnią się od procedur że pozwalają zwrócić rezultat. Deklaracja przykładowej funkcji wygląda następująco:

```
Private Function Funkcja(Parametry) As Typ
```

```
End Function
```

W parametrach określamy zmienne razem z ich typami, jako Typ końcowy określamy zmiennej jaka zostanie zwrócona przez funkcję. Przykład pokaże jak zrobić funkcję dodającą dwie liczby:

```
Private Function Dodaj(Liczba1 As Integer, Liczba2 As Integer) As Integer  
Dodaj = Liczba1 + Liczba2  
End Function
```

Następnie funkcję wywołuje się w taki prosty sposób:

```
Wynik = Dodaj(16,4)
```

Funkcja zwróci nam liczbę 20, można łatwo zauważyć że zmienną w funkcji która ma określać wynik jest jej nazwa, tak więc jeśli w funkcji o nazwie Napraw chcemy podać wynik powinno to wyglądać tak

```
Function Napraw(Co As String) As String  
Napraw = Tocomazwrócićfunkcja  
End Function
```

Oczywiście można zwracać zmienne w postaci tablicy, dzięki temu można zwrócić kilka zmiennych. Przykład funkcji liczącej średnią trzech liczb:

```
Function Srednia(Li1 As Long, Li2 As Long, Li3 As Long) As Long  
Dim Razem As Long  
Wynik = Li1 + Li2 + Li3  
Srednia = Wynik / 3  
End Function
```

## 23. Tablice

<http://visual.basic.kaem.forall.pl/>

### 1.Co to jest tablica?

Tablica fachowo jest to wydzielona część pamięci, której wielkość określa typ tej tablicy oraz ilości elementów. Wyobraźmy sobie, że chcemy zadeklarować kilka zmiennych, które będą zawierać imiona osób. Będą one tego samego typu i zawartość też będzie podobna. Zamiast deklarować każdą zmienną z osobna, możemy zadeklarować tablice o elementach różniących się tylko indeksem.

## 2. Deklarowanie tablic

Tablice deklarujemy tak jak normalne zmienne, ale do jej nazwy dodajemy liczbę elementów umieszczoną w nawiasie. Elementy liczone są od zera, chyba że pokażemy Visualowi od której liczby należy zacząć numerację. Tabela także obejmuje definiowanie jej typów oraz zakres widoczności. Przykład:

**Dim Tablica(11) As Integer**

deklaruje tablice 12 elementów o indeksach od 0 do 11. Jeśli wcześniej napiszemy

**Option Base 2**

to tablica będzie zawierała 10 elementów o indeksach od 2 do 11. Innym sposobem jest wpisanie w nawias zakresu tablicy:

**Dim Tablica(1982 To 2002) As Integer**

Będzie to tablica o indeksach liczonych od 1982 do 2001. Odwołanie do tablicy jest takie same jak dla normalnej zmiennej.

## 3. Używanie tablic

Zadeklarujmy tablice z 5 elementami

**Dim Tablica(4) As Integer**

Teraz przypiszemy każdemu elementowi z tablicy jakąś liczbę :

**Tablica(0) = 6**

**Tablica(1) = 4**

**Tablica(2) = 3**

**Tablica(3) = 9**

**Tablica(4) = 1**

Chcący wywołać wybrany element z tablicy piszemy :

**Msgbox(Tablica(2))**

Dostaniemy komunikat "3"

## 4. Co to jest macierz?

W najprostszym tłumaczeniu macierz jest to wielowymiarowa tablica. Tablica jest jednowymiarowa, "płaska". Określamy w niej tylko szerokość:

Macierz dwuwymiarowy ma natomiast określoną wysokość i szerokość:

## 5. Deklaracja macierzy

Jest bardzo podobna do deklaracji tablicy. W nawiasie podajemy liczbę elementów poszczególnych wymiarów:

**Dim Macierz (1 To 10, 25 To 28)**

I sposób odwołania się to:

A = Macierz (5,26)

## 6. Tablice dynamiczne

Opisane tu zagadnienia dotyczą też macierzy. Wyobraźmy sobie, że będzie nam potrzebna tablica, ale jeszcze nie wiemy jakiego rozmiaru (liczby elementów). Możliwe jest wtedy jej zadeklarowanie bez podawania rozmiaru:

Dim Tablica() As Integer

Teraz aby ustawić rozmiar dla tablicy używamy polecenia ReDim:

ReDim Tablica(20)

Rozmiar możemy zmieniać w dowolnym momencie

Dim Tablica() As String

ReDim Tablica(20)

ReDim Tablica(50)

ReDim Tablica(2)

Pod koniec tablica będzie miała 3 elementy (0,1,2)

Każda zmiana rozmiaru tablicy kasuje wszystkie wartości jakie tablica zawierała. Aby zachować wartości należy dodatkowo użyć parametru Preserve:

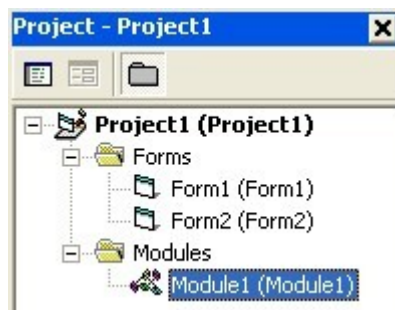
ReDim Preserve Tablica(2)

# 24. Moduły

<http://visual.basic.kaem.forall.pl>

Moduły pozwalają na tworzenie procedur/funkcji oraz deklarowanie zmiennych które będą mogły stać się dostępne dla całego projektu (różnych okien formularzy) oraz oczywiście dla samego modułu.

Moduł do projektu dodajemy przez polecenie 'Project > Add Module', po dodaniu modułu jest on dostępny w menu projektu:



Aby wybrana procedura/funkcja/zmienna zadeklarowana w module była dostępna dla całego projektu wystarczy zadeklarować ją jako Public:

Public Function Dodaj(A As Integer, B As Integer) As Integer

Dodaj = A + B

End Function



Od tej pory w każdym naszym Formie lub module możemy korzystać z wybranej funkcji. Można jeśli się chce deklarować wybrane elementy jako Public w formach, one także będą dostępne dla całego projektu, jednak nie jest to możliwe dla wszystkich elementów (np. w Formie nie da się zapisać funkcji BitBlt jako Public, co więc jeśli chcielibyśmy jej używać w innym formie ? Musielibyśmy ją deklarować jeszcze raz, takie rozwiązanie jest mało efektywne).

Co jeszcze dają nam moduły(można ich dodać dowolną ilość) ? Przedewszystkim możliwość napisania bardziej przejrzystego kodu. Można więc stworzyć sobie 3 moduły, jeden nazwać 'Funkcje Graficzne', drugi 'Funkcje Matematyczne', trzeci 'Inne'. A następnie w każdym module wpisać odpowiednie funkcje. W ten sposób nie będą one zaśmiecały głównych Form naszego projektu, który będzie od tej pory bardzo czytelny dla innych i przede wszystkim dla nas samych.

Tworzenie funkcji/procedur/zmiennych dla modułów jako Private ma sens tylko wtedy, gdy wybrane zmienne/funkcje/procedury nie wychodzą poza obręb modułu czyli są używane tylko przez moduł, przykład:

```
Private Function Dodaj(A As Integer, B As Integer) As Integer
Dodaj = A + B
End Function
```

```
Public Function Dodaj4(A As Integer, B As Integer, C As Integer, D As Integer) As Integer
Dodaj4 = Dodaj(A, B) + Dodaj(C, D)
End Function
```

Funkcja Dodaj jest dostępna tylko dla wybranego modułu, natomiast Funkcja Dodaj4 jest publiczna czyli dostępna dla wszystkich form i modułów.

## 25. Forma

<http://visual.basic.kaem.forall.pl/>

Co to za program który ma tylko jedno okienko ? Teraz nauczymy się dodawać okienka i między nimi zarządzać. Okienko dodajemy z opcji Project => Add Form. Teraz nasze nowe okienko ma nazwę Form2, możemy to oczywiście zmienić, ale na razie zostawimy Form2.

Teraz procedury formów :

Load Co - ładuje nam form do pamięci  
Co.Show - pokazuje załadowane form

Ustawmy na form1 przyciski i wpisząmy mu po kliknięciu :

```
Private Sub Command1_Click()
Load Form2
Form2.Show
End Sub
```

Po uruchomieniu programu i kliknięciu przycisku pojawi się drugie okienko.

Co.Hide - ukrywa forme

## 26. ListBox

<http://visual.basic.kaem.forall.pl/>

Obiekt ListBox jest elementem przechowującym dane tekstowe w postaci listy, z możliwością ich sortowania i układania. Właściwości jakie tu dochodzą to :

MultiSelect:

- 0 - brak zaznaczenia wielu elementów
- 1 - zaznaczenie wielu elementów przez kliknięcie
- 2 - zaznaczenie wielu elementów przez przeciąganie

Sorted:

- True - lista jest automatycznie sortowana
- False - lista nie jest segregowana

Style:

- 1 - jest to standardowy rodzaj listy
- 2 - rodzaj listy z polami checkbox

Aby dodać jakiś element do naszej listy używamy polecenia :

`List1.AddItem "dodawany tekst"`

Możemy także wybrać miejsce w którym ma się dodać nasz tekst.

`List1.AddItem "napis",0`

Zero to tutaj miejsce (linijka) w której ma być dodany nasz tekst. Obiekty z naszej listy usuwamy poleceniem :

`List1.RemoveItem 0`

Usuwa to wybraną linie z listy, oczywiście 1 linia ma wartość 0, druga to 1 itd.. Jeżeli chcemy usunąć nie istniejącą linie program wyrzuci błąd. Tak samo stanie się jeśli chcemy dodać element w zbyt dalekie miejsce. Można temu zapobiec stosując zmienną przechowującą ilość elementów w liście.

```
If List1.ListCount > -1 Then  
List1.RemoveItem 0  
End If
```

List1.ListCount to właśnie ilość naszych elementów. Całą listę możemy wyczyścić poleceniem :

`List1.Clear`

Teraz czas na przechwytywanie zaznaczonego elementu. Jest to banalne, połączmy na formę Label i Listę, dodajmy do listy z 4 elementy, kliknijmy na nią dwukrotnie i napiszmy :

`Private Sub List1_Click()`

```
Label1.Caption = List1.Text  
End Sub
```

List1.Text przechowuje treść zaznaczonego elementu. Kolejną rzeczą jest zmienna przechowująca który element został zaznaczony.

```
Private Sub List1_Click()  
Label1.Caption = List1.ListIndex  
End Sub
```

Jeżeli żaden element listy nie jest zaznaczony to List1.ListIndex przechowuje wartość -1. Możemy także sami (bez zaznaczenia przez użytkownika) wywołać tekst wybranego elementu.

List1.List(0)

Wyświetli nam tekst pierwszego elementu. Dla lepszego zrozumienia mały przykładzik :

Na formie umieszczamy List1, Label1, Label2, Command1, Command2, Command3 i Text1. Poniższy kod zostanie wyjaśniony liniami :

```
01. Private Sub Form_Load()  
02. Command1.Caption = "usuń"  
03. Command2.Caption = "dodaj"  
04. Command3.Caption = "odśwież"  
05. List1.AddItem ("Visual")  
06. List1.AddItem ("Basic")  
07. List1.AddItem ("Test")  
08. Label1.Caption = "Elementów : " & List1.ListCount  
09. End Sub  
10. Private Sub Command1_Click()  
11. If (List1.ListIndex > -1) Then  
12. List1.RemoveItem (List1.ListIndex)  
13. End If  
14. End Sub  
15. Private Sub Command2_Click()  
16. If Text1.Text <> "" Then  
17. List1.AddItem Text1.Text  
18. End If  
19. End Sub  
20. Private Sub Command3_Click()  
21. Label1.Caption = "Elementów : " & List1.ListCount  
22. End Sub  
23. Private Sub List1_Click()  
24. Label2.Caption = List1.Text  
25. End Sub
```

Linia 5,6,7 - dodaj do listy 3 elementy ("Visual", "Basic", "Test")

Linia 8 - wyświetla na Label1 ilość elementów listy

Linia 10 - jeżeli klikłeś w przycisk 1

Linia 11 - i jest jakiś zaznaczony element na liście to

Linia 12 - usuń z listy zaznaczony element

Linia 15 - jeżeli klikłeś w drugi przycisk

Linia 16 - i pole tekstowe nie jest puste to

Linia 17 - dodaj do listy element o teksie z pola Text1

Linia 20 - jeżeli klikłeś w trzeci przycisk to

Linia 21 - odśwież Label1 i wpisz w nim aktualną liczbę elementów

Linia 23 - jeżeli klikłeś w listę to  
Linia 24 - wypisz kliknięty tekst w Label2

## 27. DriveListBox, DirListBox, FileListBox

<http://visual.basic.kaem.forall.pl/>

Obsługa tych kontroltek jest bardzo prosta, służą one do listowania plików i folderów. Umieścimy po prostu te trzy elementy na formie i włączmy program. Można już łatwo poruszać się po folderach i plikach. Każdy z tych elementów posiada właściwość o aktualnej ścieżce. Są to :

Dla DriveListBox : Drive1.Drive  
Dla DirListBox : Dir1.Path  
Dla FileListBox : File1.Path

Dzięki temu możemy zgrać ze sobą każdą z tych kontroltek. Klikamy na Drive1 (Drive1\_Change()) i piszemy

```
Dir1.Path = Drive1.Drive
```

Oznacza to, że po zmianie dysku (Drive1) program ma uaktualnić listę folderów (Dir1). Jeszcze tylko zapobieganie błędowi podczas gdy wybrany hdd/cd /a/... są puste, użyjemy tu wyłączenie błędów (poczytasz o tym w dziale obsługa błędów).

```
On Error Resume Next  
Dir1.Path = Drive1.Drive
```

Zgrajmy też wybór folderów z listą plików, kliknijmy na Dir1 i wystukajmy :

```
File1.Path = Dir1.Path
```

Aby w FileListBox wyświetlić tylko pliki pożądanego typu musimy zmienić właściwość Pattern z \*.\* na np. \*.jpg. Pliki z FileListBox można zaznaczyć i uzyskać całkowitą ścieżkę do naszego zaznaczonego pliku.

```
Private Sub File1_Click()  
Msgbox File1.Path & "\" & File1.FileName  
End Sub
```

File1.Path to ścieżka do pliku, a File1.FileName to nazwa pliku wraz z rozszerzeniem. Po uruchomieniu programu i sprawdzeniu ścieżek dla plików okazuje się, że pliki z katalogów dyskowych mają przy sobie dwa backslashe (np. : C:\autoexec.bat). Używając poznanej wcześniej funkcji Len możemy temu zapobiec :

```
Private Sub File1_Click()  
bslash = "\"  
If Len(File1.Path) < 4 Then bslash = ""  
MsgBox File1.Path & bslash & File1.FileName  
End Sub
```

Teraz program sprawdza czy plik znajduje się w katalogu dyskowym (bezpośrednio w C:\, D:\...), jeżeli ten plik tam jest to backslash zostaje usunięty.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 28. Timer

<http://visual.basic.kaem.forall.pl/>

Pośród komponentów znajduje się dotyczący tej lekcji element (Timer), służy on do wywoływania różnych czynności w wyznaczonych przez nas odstępach czasu. Odstępy te są wyrażane w milisekundach (1 sekunda = 1000 milisekund). Aby nasz timer zaczął coś robić musimy mu ustawić czas we właściwości Interval, ustawmy przykładowo 5 sekund. Dwukrotnie klikając w nasz timer ustawiamy akcje, stwórzmy wszystko by wyglądało tak:

```
Private Sub Timer1_Timer()  
MsgBox ("???)  
End Sub
```

Program powinien co 5 sekund wyświetlać komunikat "???". W kolejnym przykładzie umieścimy na formie label i timer, timerowi ustawiamy Interval na 1000, klikamy w niego i piszemy :

```
Public A As Integer  
Private Sub Timer1_Timer()  
A = A + 1  
Label1.Caption = "Czas włączenia programu (w sekundach) : " & A  
End Sub
```

Aby wyłączyć lub włączyć nasz timer nadajemy mu Enabled odpowiednio na Flase lub True

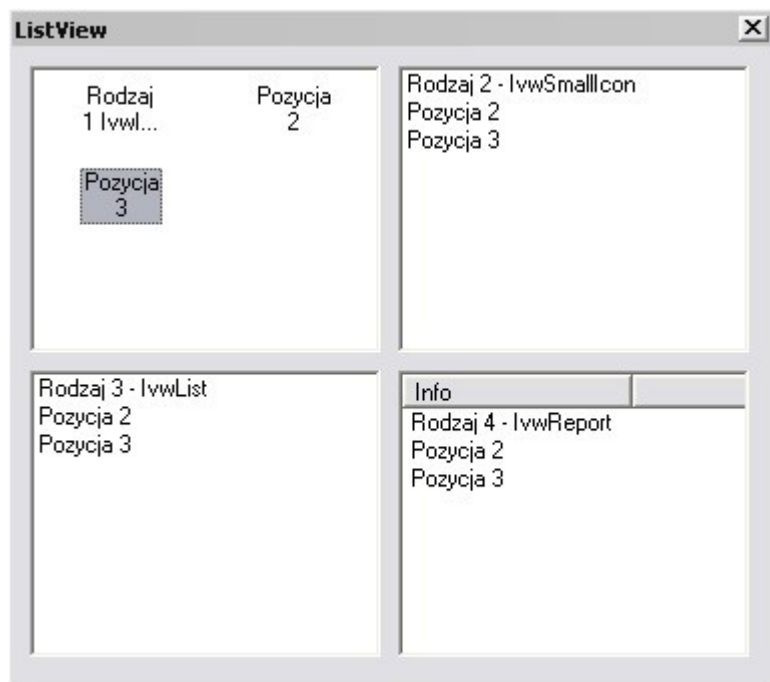
## 29. ListView

<http://visual.basic.kaem.forall.pl/>

ListView jest obiektem dodatkowym, aby móc go używać musimy z Komponentów (Ctrl+T) wybrać Microsoft Windows Common Controls 5.0 (SP2). Używając jej wypadło by też dołączać do naszego .exe odpowiedni plik .ocx (comctl32.ocx)

### 1. Ogólnie

ListView pozwala na wyświetlanie listy w 4 różnych trybach (jak na obrazku)



Tryb wyświetlania elementów wybieramy przez właściwość `.View` przypisując jej jcyfre od 1 do 4

Elementem który wyróżnia `ListView` jest łatwa obsługa oraz możliwość dodania do dowolnego typu listy obrazków.

Aby dodać wybrany element do listy należy użyć prostego polecenia:

`ListView1.ListItems.Add Index,Klucz,Tekst,Obrazek,Ikona`

Pola `Index` (pozycja w której chcemy dodać tekst), `Klucz`, `Obrazek` oraz `Ikona` nie są polami obowiązkowymi do dodania, najprostrze dodanie elementu można więc zapisać w taki sposób:

`ListView1.ListItems.Add , , "Tekst"`

Pozycje z listy możemy łatwo usunąć stosując:

`ListView1.ListItems.Remove Index`

`Index` jest pozycją naszego elementu w liście. Listę czyścimy przez:

`ListView1.ListItems.Clear`

Ważna jest też dla nas liczba elementów z listy:

`ListView1.ListItems.Count`

numer oraz tekst zaznaczonego elementu:

`ListView1.SelectedItem.Index`

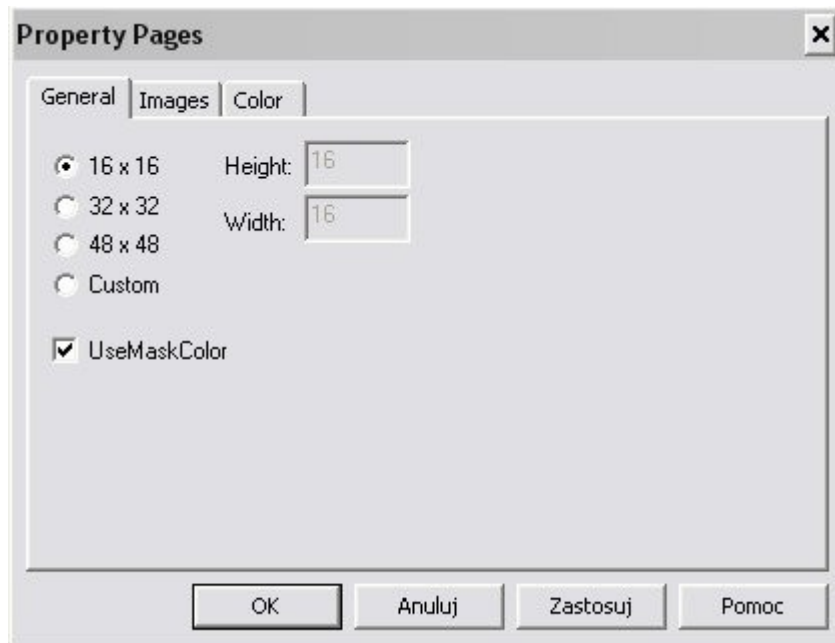
`ListView1.SelectedItem.Text`

oraz tekst dowolnego obiektu:

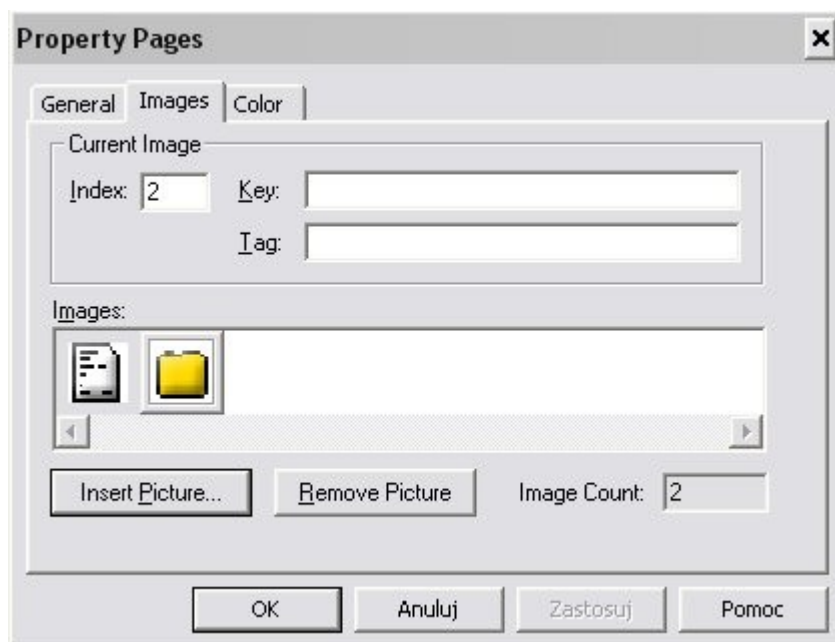
`MsgBox ListView1.ListItems(Index).Text`

## 2. Dodawanie obrazków

Jeżeli chcemy dodać obrazki do naszej listy będziemy potrzebowali użyć obiektu `ImageList` (jest także w Microsoft Windows Common Controls 5.0 (SP2)). Stawiamy ten element na naszej formie, klikamy na niego prawym przyciskiem myszki i wybieramy 'Properties'. Teraz wybieramy rozmiar obrazków (przykładowo wybierzmy 16x16 aby obrazki pasowały do 2,3 i 4 sposobu wyświetlania listy)



W zakładce Image możemy dodać dowolną ilość obrazków



Każdy dodany przez nas obrazek posiada odpowiedni numer (Index), który będziemy potem wykorzystywali przy `ListView`. Teraz jedyne co należy zrobić to przypisać `ImageList` do `ListView`, małe ikony (16x16) przypisujemy przez `SmallIcons`, duże przez samo `Icons`:

```
ListView1.SmallIcons = ImageList1
```

Gdy nasze małe ikony są już przypisane można ich używać:

```
ListView1.ListItems.Add , , "Tekst 1", , 2  
ListView1.ListItems.Add , , "Tekst 2", , 1  
ListView1.ListItems.Add , , "Tekst 3", , 1
```



### 3. Typ 4 - IvwReport

Ten typ wyświetlania listy jest nieco bardziej skomplikowany jednak pozwala na o wiele więcej. Używając go przedewszystkim należy zadeklarować ilość kolumn, tekst na nich oraz ich szerokość.

```
ListView1.ColumnHeaders.Add Index, Key, Text, Szerokosc, PozycjaTekstu
```

Dodanie 3 kolumn wyglądało by mniej więcej tak:

```
ListView1.ColumnHeaders.Add , , "Nazwa", 3000  
ListView1.ColumnHeaders.Add , , "Info", 1000  
ListView1.ColumnHeaders.Add , , "Rozmiar", 1000, 1
```

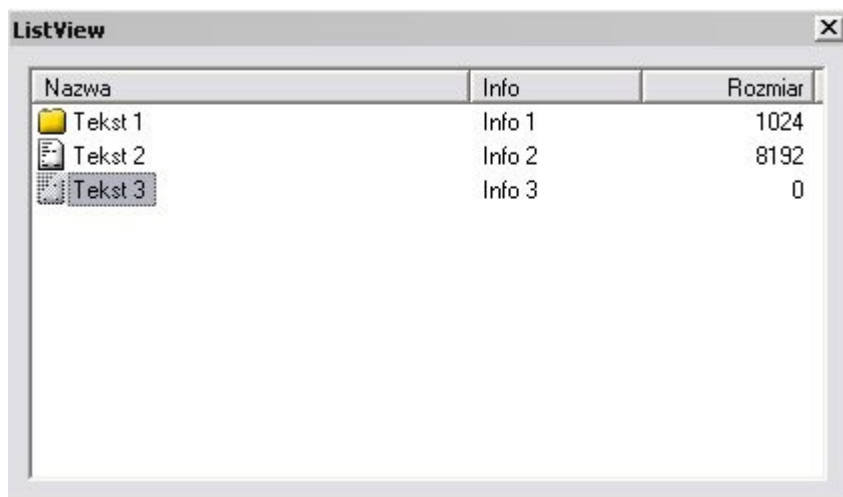
Rozmiar wynosi 3000,1000,1000 ponieważ użyłem ScaleMode = 1 i podałem go w Twipsach, PozycjaTekstu może przyjmować 3 wartości

- 0 - do lewej strony
- 1 - do prawej strony
- 2 - wyśrodkowane

Elementy dodajemy w taki sam sposób, jednak jeżeli chcemy uzupełnić kolejne komórki listy należy zrobić to trochę inaczej. Należy przypisać do zmiennej komórkę do którego chcemy coś dodać:

```
Set Komorka = ListView1.ListItems.Add( , "Tekst 1", , 2)  
Komorka.SubItems(1) = "A"  
Komorka.SubItems(2) = "B"
```





Wyciągnięcie elementów z takiej listy jest już o wiele prostsze, pierwszy element pobieramy w normalny sposób:

`ListView1.SelectedItem.Text`

Elementy z następnych kolumn w sposób następujący:

`ListView1.SelectedItem.SubItems(Index)`

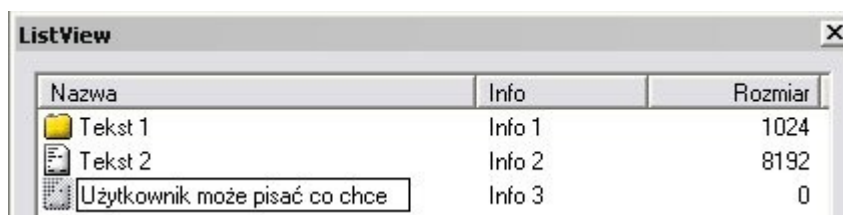
W przypadku gdy chcemy pobrać informacje z innego niż zaznaczonego elementu postępujemy analogicznie:

`A = ListView1.ListItems(Index).Text`

`B = ListView1.ListItems(Index).SubItems(Index2)`

Index to numer elementu, Index2 to index kolumny.

Warto jeszcze zwrócić uwagę na to że użytkownik może po dwu/trzy krotnym kliknięciu na wybrany element edytować go



Decyduje o tym właściwość `LabelWrap`, aby wyłączyć użytkownikowi taką możliwość wystarczy ustawić `ListView1.LabelWrap = False`

Możemy także sami włączyć edycję zaznaczonego elementu przez `ListView1.StartLabelEdit`

#### 4. Zdarzenia i metody

Zdarzenia:

`AfterLabelEdit(Cancel As Integer, NewString As String)` - możemy dzięki temu sprawdzić jak użytkownik zedytował wybrane pole, `NewString` to nowa nazwa wpisana przez użytkownika, gdy chcemy pobrać pierwotną nazwę wybieramy po prostu nazwę

edytowanej kolumny (w tym zdarzeniu ma jeszcze starą wartość)

**ListView1\_ColumnClick(ByVal ColumnHeader As ComctlLib.ColumnHeader)** - zdarzenie zachodzi po kliknięciu w wybraną kolumnę (w 4 typie wyświetlania listy) można dzięki temu uzyskać tekst z klikniętej kolumny a co za tym idzie jej index. Jeżeli nadamy kolumną przy ich tworzeniu wartości Key będziemy mogli stworzyć sortowanie listy według klucza (numeru kolumny) w którą klikneliśmy

Metody:

**ListView1.FindItem** - pozwala zwrócić tekst wiersza po jego nazwie, można używać gdy nie wiemy jaki index ma wybrane przez nas pole

**ListView1.GetFirstVisible** - zwraca tekst z pierwszego widocznego wiersza

Właściwości:

**ListView1.MultiSelect** - ustawione na true pozwala użytkownikowi zaznaczyć więcej niż jeden obiekt na liście. Wyświetlenie **ListView1.ListItems.Selected.Text** wyświetli tylko ostatnio zaznaczony element. Żeby sprawdzić jakie obiekty są zaznaczone należy użyć pętli która sprawdza po kolei jaki obiekt jest zaznaczony

**Dim A As Long**

**Do While A < ListView1.ListItems.Count**

**A = A + 1**

**Sprawdz = ListView1.ListItems(A).Selected**

**If Sprawdz = True Then MsgBox "Zaznaczony: " & ListView1.ListItems(A).Text**

**Loop**

**ListView1.ListItems(A).Selected** przyjmuje True lub False w zależności czy obiekt o indeksie A jest zaznaczony lub nie

**ListView1.StartLabelEdit** - pozwala na edycję wybranego pola w wybranym przez nas momencie. W przypadku kliknięcia w button lub innego zdarzenia odbierającego fokus należy najpierw zwrócić fokus dla ListView:

**Private Sub Command1\_Click()**

**ListView1.SetFocus**

**ListView1.StartLabelEdit**

**End Sub**

## 5. Poprawiamy zaznaczanie elementu w lvwReport

Używając wyświetlania listy typu **lvwReport** można od razu zauważyć że zaznaczanie elementów nie wygląda w taki sposób jakbyśmy tego chcieli, kliknięcie w tekst przy innej kolumnie, lub po prawej obok tekstu który chcemy zaznaczyć nie działa poprawnie. Wiersz jest zaznaczany tylko wtedy gdy klikniemy na tekst/obrazek z pierwszej kolumny. Wypadało by to poprawić. Standardowo szerokość wiersza w polu listy powinna wynosić 17 pikseli. Musimy użyć zdarzenia **ListView1\_MouseUp** zwraca nam ono pozycję kliknięcia myszy w twipsach. Dzięki temu możemy sprawdzić gdzie kliknął użytkownik i które pole zaznaczyć.

**Private Sub ListView1\_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)**

**y = (y / 15 - (y / 15 Mod 17)) / 17**

**ListView1.ListItems(y).Selected = True**

**End Sub**

y obliczamy przez podzielenie y przez 15 ( zamieniamy twipsy na pixele, następnie odejmujemy resztę z dzielenia przez 17 i całość dzielimy przez 17. Otrzymamy dzięki temu liczbę całkowitą która informuje nas o numerze elementu który został kliknięty. Można także zrobić zaznaczanie elementów dodając MouseDown, będzie to wyglądało efektywniej. Opisany powyżej kod musi zostać poprawiony, nie będzie on działał prawidłowo jeżeli klikniemy w miejscu np. 6 elementu a w liście będzie ich tylko 4, pod obliczeniem y dopisujemy:

```
If ListView1.ListItems.Count >= y Then  
ListView1.ListItems(y).Selected = True  
End If
```

Kod powinien działać poprawnie... jednak tak nie jest, nie zostało jeszcze uwzględnione to, ile wierszy jest uwzględnione. Pozycja y nie zwróci nam indexu wiersza tylko jedno z miejsc z listy na które klikneliśmy, a co jeżeli pewna część elementów będzie niewidoczna (lista będzie przewinięta suwakiem w dół) ? W takim wypadku zaznaczony element nie będzie prawidłowy, trzeba napisać do tego poprawkę uwzględniającą przewinięte kolumny:

```
Private Sub ListView1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)  
y = (y / 15 - (y / 15 Mod 17)) / 17  
Nazwa = ListView1.GetFirstVisible  
  
Dim A As Long  
Do While A < ListView1.ListItems.Count  
A = A + 1  
If Nazwa = ListView1.ListItems(A).Text Then GoTo Koniec  
Loop  
  
Koniec:  
y = y + A - 1  
  
If ListView1.ListItems.Count >= y Then  
ListView1.ListItems(y).Selected = True  
End If  
End Sub
```

Oczywiście to nie ma mniejszego sensu gdy wartości z ListItems nie są pozycjami kluczami (powtarza się np. dwa razy ten sam tekst).

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (3)

## 30. TreeView

<http://visual.basic.kaem.forall.pl/>

TreeView pozwala wyświetlać listę obiektów w postaci drzewa:



Aby móc korzystać z tego obiektu należy z komponentów (Ctrl+T) wybrać 'Microsoft Windows Common Controls 5.0 (SP2)



Obiekty dodajemy do drzewa przez funkcję Nodes

```
TreeView1.Nodes.Add
[Relacja],[TypRelacji],[Klucz],[WyświetlanyTekst],[Obrazek],[ZaznaczonyObrazek]
```

Dodanie kilku obiektów:

```
TreeView1.Nodes.Add , , "Tekst 1"
TreeView1.Nodes.Add , , "Tekst 2"
TreeView1.Nodes.Add , , "Tekst 3"
TreeView1.Nodes.Add , , "Tekst 4"
```



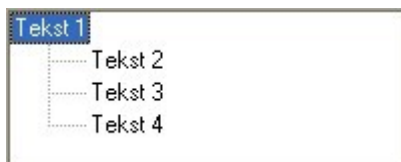
Dodanie pod obiektów do listy jest proste, po pierwsze musimy nadać jakiś Klucz obiektowi matce:

```
TreeView1.Nodes.Add , , "Matka", "Tekst 1"
```

Teraz aby dodać pod obiekt który będzie rozwinięciem Matki należy użyć Relacji jako klucza i podać typ relacji:

```
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 2"
```

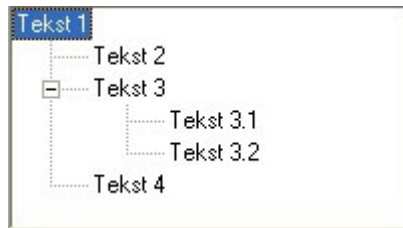
```
TreeView1.Nodes.Add , , "Matka", "Tekst 1"
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 2"
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 3"
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 4"
```



Każdy z pod elementów także może być matką innych elementów, wystarczy nadać mu klucz

```
TreeView1.Nodes.Add , , "Matka", "Tekst 1"
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 2"
TreeView1.Nodes.Add "Matka", tvwChild, "TezJestemMatka", "Tekst 3"
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 4"
```

```
TreeView1.Nodes.Add "TezJestemMatka", tvwChild, , "Tekst 3.1"
TreeView1.Nodes.Add "TezJestemMatka", tvwChild, , "Tekst 3.2"
```



Elementy z listy są standardowo 'zwinięte' aby rozwinąć listę należy użyć EnsureVisible

```
TreeView1.Nodes.Add , , "Matka", "Tekst 1"
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 2"
TreeView1.Nodes.Add "Matka", tvwChild, "TezJestemMatka", "Tekst 3"
TreeView1.Nodes.Add "Matka", tvwChild, , "Tekst 4"
TreeView1.Nodes.Add "TezJestemMatka", tvwChild, "PokazMnie", "Tekst 3.1"
TreeView1.Nodes.Add "TezJestemMatka", tvwChild, , "Tekst 3.2"
TreeView1.Nodes("PokazMnie").EnsureVisible
```

Opcje:

**TreeView1.Nodes.Remove (Index)** - usuwa wybraną pozycję z drzewa o wybranym Indeksie lub kluczu:

```
TreeView1.Nodes.Remove (4) - usunie 4 element z listy
TreeView1.Nodes.Remove ("TezJestemMatka") - usunie element o kluczu
'TezJestemMatka'
```

**TreeView1.Nodes.Clear** - czyści całe drzewo

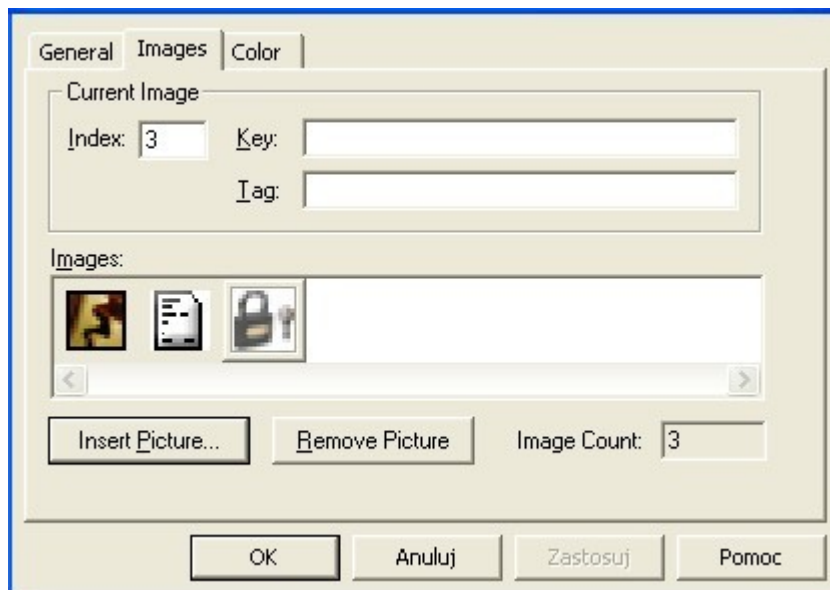
**TreeView1.SelectedItem.Text** - przechowuje tekst zaznaczonego elementu

## Obrazki

Obrazki dodajemy na prawie identycznej zasadzie jak do obiektu ListView. Najpierw umieścimy na formie obiekt ImageList



Teraz kliknijmy na ImageList1 prawym myszki i wybierzmy Properties. W zakładce General ustawmy wielkość obrazków na 16x16, w Images dodajmy kilka przygotowanych przez nas obrazków



Aby obrazki były powiązane z TreeView trzeba użyć przyporządkowania:

`TreeView1.ImageList = ImageList1`

Gdy przyporządkowanie jest poprawne można używać indeksów obrazków z ImageList w miejsca [Obrazek],[ZaznaczonyObrazek] podczas dodawania elementów

`TreeView1.Nodes.Add  
[Relacja],[TypRelacji],[Klucz],[WyświetlanyTekst],[Obrazek],[ZaznaczonyObrazek]`

**Przykład:**

```
TreeView1.ImageList = ImageList1
TreeView1.Nodes.Add , , "glowne", "Ustawienia", 1
TreeView1.Nodes.Add "glowne", twwChild, , "Opcje tekstu", 2
TreeView1.Nodes.Add "glowne", twwChild, , "Format", 2
TreeView1.Nodes.Add "glowne", twwChild, , "Widok", 2
TreeView1.Nodes.Add "glowne", twwChild, "rozne", "Różne", 1
TreeView1.Nodes.Add "rozne", twwChild, , "Czas", 2
TreeView1.Nodes.Add "rozne", twwChild, "pokaz", "Hasło", 3
TreeView1.Nodes("pokaz").EnsureVisible
```



# 31. Dostęp do plików

<http://visual.basic.kaem.forall.pl/>

W Visual Basic-u operacje takie jak zapisywanie informacji do pliku, odczytywanie ich i dopisywanie to kilka prościutkich poleceń. Plik możemy otworzyć na 3 sposoby :

Input - odczytanie

Append - dopisywanie do pliku

Output - nadpisywanie lub w przypadku nie istnienia pliku tworzenie go, a w przypadku istnienia kasuje jego zawartość

Aby zdefiniować programowi który plik ma być edytowany i w jaki sposób należy posłużyć się procedurą Open.

Open plik For Typ As #liczba

Plik to ścieżka dostępu do pliku, np. : "C:\aaa.txt"

Typ to jeden z trzech typów dostępu do pliku.

Liczba to numer pliku, jest nam to potrzebne gdy chcemy edytować kilka plików na raz.

Przykład :

Open "C:\aaa.txt" For Output As #1

Tym poleceniem otworzymy plik aaa.txt i ustawimy go do pisania w nim tekstu. Kiedy chcemy otworzyć następny plik to musimy albo użyć jako liczby #1, albo zamknąć uprzednio otwarty plik, zamykamy go poleceniem Close.

Open "C:\aaa.txt" For Output As #1

Close #1

Poleceniem Print wpisujemy do pliku nasz tekst, możemy to robić tylko w przypadku otwarcia pliku jako Append i Output :

Open "C:\aaa.txt" For Output As #1

Print #1,"Tekst wpisany do pliku"

Close #1

Chcąc teraz odczytać tekst z pliku musimy otworzyć go jako Input i użyć polecenia Line Input :

Open "C:\aaa.txt" For Input As #1

Line Input #1, a

Close #1

Teraz nasza zmienna a zawiera pierwszą linijkę z pliku, czyli "Tekst wpisany do pliku !". Oczywiście sprawą jest fakt, że możemy dodawać dużo linijek do pliku :

Open "C:\aaa.txt" For Output As #1

Print #1,"1 Linijka"

Print #1,"2 Linijka"

Print #1,"3 Linijka"

Close #1

Warto zauważyć że zapisywanie danych do pliku za pomocą Print automatycznie przechodzi do nowej linii w pliku. Jeżeli chcemy aby program nie przechodził do nowej linii

należy na końcu linijki z procedurą Print postawić średnik

```
Open "C:\aaa.txt" For Output As #1
Print #1,"To wszystko ";
Print #1,"będzie w ";
Print #1,"jednej lini !";
Close #1
```

To samo tyczy się odczytywania z pliku. Możemy też zapisywać (Write) wiele wyrażeń do jednej linii a następnie je odczytywać (Input), zapis wygląda tak :

```
Open "C:\aaa.txt" For Output As #1
Write #1,"Imie","Nazwisko","Fach"
Write #1,"Henryk","Sienkiewicz","Pisarz"
Close #1
```

Natomiast odczyt tak :

```
Open "C:\aaa.txt" For Input As #1
Input #1,a,b,c
Input #1,d,e,f
Close #1
```

Nasze zmienne a, b, c, d, e, f przechowują kolejne dane z pliku, daje to duże możliwości podczas robienia i zapisywania baz danych, możemy użyć tutaj także dwuwymiarowych tablic. Czasem zdarza się, że chcemy odczytać jakiś plik, ale nie wiemy ile ma linijek. Tutaj stosujemy funkcję EOF która zwraca 2 wartości, True jeśli plik nie ma już więcej linijek do odczytania i False gdy plik nie jest jeszcze odczytany do końca. Aby było łatwiej przyswoić tą funkcję przykładzik. Umieścimy na formie ListBox, stwórzmy jakiś plik w C:\ i wpiszmy do niego trochę bzdur (tak z 5 - 10 linijek). Teraz napiszemy program który odczytuje linijki z tego pliku i wpisuje je do pola listy :

```
1. Private Sub Form_Load()
2. Open "C:\aaa.txt" For Input As #1
3. Do While EOF(1) = False
4. Line Input #1, a
5. List1.AddItem a
6. Loop
7. Close #1
8. End Sub
```

Linia 2 - otwiera plik aaa.txt do odczytu jako 1  
Linia 3 - powtarzaj pętlę dopóki koniec z pliku 1 jest fałszem  
Linia 4 - odczytaj linie z pliku  
Linia 5 - dodaj odczytaną linie do listy  
Linia 6 - Koniec pętli  
Linia 7 - Zamknij plik 1

## 32. Operacje na plikach

<http://visual.basic.kaem.forall.pl/>

Umiemy już zapisywać i odczytywać tekst z plików, teraz nauczymy się te pliki przenosić kopiować, zmieniać nazwy, poznamy też operacje na katalogach.



## Kill

Usuwa plik, w przypadku, gdy plik nie istnieje zwróci nam błąd.

```
Kill "C:\aaa.txt"
```

Usunie plik aaa.txt

```
Kill "C:\aaa\*.*)"
```

Usunie wszystkie pliki z katalogu aaa (zostawi jednak foldery)

## Name

```
Name "C:\aaa.txt" As "C:\bbb.txt"
```

Zmieni nazwe naszego pliku na bbb.txt

```
Name "C:\aaa.txt" As "C:\aaa\aaa.txt"
```

Przeniesie nasz plik do folderu aaa

## FileCopy

Kopiuje nasz plik

```
FileCopy "C:\aaa.txt", "C:\aaa\bbb.txt"
```

## FileLen

Sprawdza wielkość pliku w bajtach.

```
Bajty = FileLen("C:\aaa.txt")
```

# 33. Operacje na katalogach

<http://visual.basic.kaem.forall.pl/>

## RmDir

Usuwa wybrany katalog

```
RmDir "C:\aaa\"
```

## Mkdir

Tworzy nam katalog

```
Mkdir "C:\aaa\"
```

## Name

```
Name "C:\aaa\" As "C:\bbb\"
```

Zmieni nazwę folderu z aaa na bbb, zachowując wszystkie pliki wewnątrz

## 34. Atrybuty plików i folderów

<http://visual.basic.kaem.forall.pl/>

Atrybuty dla plików i folderów najlepiej pobierać przez

Funkcje uzyskiwania tych atrybutów należy zadeklarować:

```
Private Declare Function GetFileAttributes Lib "kernel32" Alias "GetFileAttributesA" (ByVal  
lpFileName As String) As Long
```

```
Private Declare Function SetFileAttributes Lib "kernel32" Alias "SetFileAttributesA" (ByVal  
lpFileName As String, ByVal dwFileAttributes As Long) As Long
```

Później aby użyć atrybut pliku należy użyć zadeklarowanej funkcji GetFileAttributes:

```
Atrybuty = GetFileAttributes("AdresFolderuLubPliku")
```

Aby ustawić wybrany atrybut należy użyć drugiej funkcji:

```
SetFileAttributes "PlikLubFolder",Atrybuty
```

Wartość atrybutów to wartość liczbowa, składa się z następujących wartości:

- 1 - tylko do odczytu
- 2 - ukryty
- 4 - systemowy
- 16 - katalog
- 32 - archiwalny
- 128 - brak atrybutów
- 160 - tymczasowy

Gdy chcemy uzyskać wartość mieszaną poprostu dodajemy wybrane przez nas wartości atrybutów, przykładowo dla pliku który ma być systemowy i ukryty należało by dać atrybut o wartości 6 (2 + 4), dla ukrytego folderu który jest tylko do odczytu wartość 19 (2 + 16 + 1).

Identycznie ma się pobieranie atrybutów, należy sprawdzić z jakich liczb składa się pobrany przez nas atrybut.

```
Dim Atrybuty As Byte  
Atrybuty = GetFileAttributes("C:/")
```

```
If Atrybuty > 159 Then  
MsgBox "Tymczasowy"  
Atrybuty = Atrybuty - 160  
End If
```

```
If Atrybuty > 127 Then  
MsgBox "Brak Atrybutów"  
Atrybuty = Atrybuty - 128  
End If
```

```
If Atrybuty > 31 Then
MsgBox "Archiwalny"
Atrybuty = Atrybuty - 32
End If
```

```
If Atrybuty > 15 Then
MsgBox "Katalog"
Atrybuty = Atrybuty - 16
End If
```

```
If Atrybuty > 3 Then
MsgBox "Systemowy"
Atrybuty = Atrybuty - 4
End If
```

```
If Atrybuty > 1 Then
MsgBox "Ukryty"
Atrybuty = Atrybuty - 2
End If
```

```
If Atrybuty > 0 Then
MsgBox "Tylko do odczytu"
End If
```

Oczywiście lepiej było by to zrobić na instrukcji Case jednak pokazanie tego na If pozwoli łatwiej to zrozumieć.

## 35. Spis plików i katalogów

<http://visual.basic.kaem.forall.pl/>

W tym dziale opisze jak wyciągnąć za pomocą API nazwy plików oraz katalogów z wybranego katalogu. Jest to o wiele szybszy sposób niż wyciąganie nazw plików i folderów z umieszczonych na formie kontrolki FileListBox i DirListBox.

Mając doczynienia z API musimy zadeklarować pewne procedury oraz typy zmiennych:

```
Private Type FILETIME
dwLowDateTime As Long
dwHighDateTime As Long
End Type
```

```
Private Type WIN32_FIND_DATA
Atrybuty As Long
ftCreationTime As FILETIME
ftLastAccessTime As FILETIME
ftLastWriteTime As FILETIME
RozmiarNaHDD As Long
Rozmiar As Long
dwReserved0 As Long
dwReserved1 As Long
Nazwa As String * 260
cAlternate As String * 14
End Type
```

Private WFD As WIN32\_FIND\_DATA

```
Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" (ByVal  
lpFileName As String, lpFindFileData As WIN32_FIND_DATA) As Long  
Private Declare Function FindNextFile Lib "kernel32" Alias "FindNextFileA" (ByVal FindFile  
As Long, lpFindFileData As WIN32_FIND_DATA) As Long
```

Po takim zadeklarowaniu można już korzystać z dwóch nowych funkcji: FindFirstFile, oraz FindNextFile.

Pierwsza funkcja znajduje 1 plik w wybranym katalogu i tworzy coś w rodzaju 'uchwyty' dla drugiej funkcji. Druga funkcja wyszukuje kolejne pliki pobierając 'uchwyt' z funkcji poprzedniej.

Wykorzystanie jest proste:

```
Uchwyt = FindFirstFile("C:\*", WFD)
```

Od tej pory zmienna WFD zawiera różne wartości o pierwszym pliku/folderze z wybranego folderu. Do najważniejszych własności należą:

```
WFD.Nazwa  
WFD.Rozmiar  
WFD.RozmiarNaHDD  
WFD.Atrybuty
```

Żeby określić czy pobrany przez nas plik/folder jest odpowiednio plikiem lub folderem musimy sprawdzić to sprawdzając jego atrybuty. ( WFD.Atrybuty ), opisane jest to w poprzedniej lekcji.

Chcąc uzyskać następny element(y) używamy drugiej funkcji, tym razem nie musimy już przypisywać jej wyniku do uchwytu, jednak przydała by się zmienna zwracająca czy istnieją jakieś pliki/foldery w wybranym katalogu czy nie:

```
Istnieje = FindNextFile(Uchwyt, WFD)
```

Aby pobrać informacje o wszystkich plikach z wybranego katalogu należało by więc stworzyć pętlę która powtarza się dopóki istnieje wartość 'Istnieje'.

```
Dim Uchwyt As Long  
Dim Istnieje As Long  
Istnieje = 1
```

```
Uchwyt = FindFirstFile("C:\*", WFD)
```

```
Do While Istnieje  
MsgBox WFD.Nazwa  
Istnieje = FindNextFile(Uchwyt, WFD)  
Loop
```

Kod powyższej funkcji wyświetli wszystkie elementy z katalogu C:\. Oprócz WFD.Nazwa możemy także wyświetlić inne opisane wcześniej właściwości pliku/katalogu.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (3)

## 36. Obsługa błędów

<http://visual.basic.kaem.forall.pl/>

Często zdarza się, że w programie występuje jakiś błąd, po błędzie program zawiesza się i wyłącza. Błędowi można jednak zapobiegać (raczej naprawiać w trakcie działania programu). Aby usunąć wyłączenie komunikatu po błędzie piszemy w jakiejś procedurze:

### On Error Resume Next

Polecenie to wyłącza błędy i po wystąpieniu jakiegogoś programu działa dalej. Możemy także dowiedzieć się jaki błąd wystąpił i jemu zapobiec, pokaże to na przykładzie z błędem dzielenia przez 0 :

#### On Error Resume Next

```
A = 6 / 0
If Err.Number > 0 Then GoTo Bład
If Err.Number = 0 Then GoTo Bez
Bład:
MsgBox ("Wystąpił błąd : " & Err.Number)
Exit Sub
Bez:
MsgBox ("Ok. nie ma błędu")
```

Zmienna Err.Number przechowuje numer błędu, natomiast zmienna Err.Description jego komunikat, gdy numer ma wartość 0 to błędu nie ma, analizując to gdy jest błąd program idzie do miejsca Bład:, potem wyświetla komunikat i funkcją Exit Sub zamyka procedurę. Gdy błędu zaś nie ma to program przechodzi do miejsca Bez: pomijając Exit Sub. Dzielenie przez 0 to błąd nr. 11, stwórzmy na formie pole TextBox (nazwa domyślna Text1), pole Command (Command1), kliknijmy w przycisk i wpiszmy coś takiego :

#### On Error Resume Next

```
A = 6 / Text1.Text
If Err.Number > 0 Then GoTo Bład
If Err.Number = 0 Then GoTo Bez
Bład:
If Err.Number = 11 Then
MsgBox ("Błąd dzielenia przez 0")
Else: MsgBox ("Wystąpił nieznan błąd : " & Err.Number)
End If
Exit Sub
Bez:
MsgBox ("Ok. nie ma błędu")
```

Wpiszmy teraz w pole tekstowe 0 i kliknijmy przycisk, powinno nam wyskoczyć " Błąd dzielenia przez 0 ", wpiszmy teraz np. 1, powinno być "Ok. nie ma błędu". Teraz wpiszmy jakąś literę np. a. Dostaniemy komunikat "Wystąpił nieznan błąd : 13". Jest to błąd dzielenia przez znak, możemy na niego też zrobić komunikat : If Err.Number = 13 Then MsgBox("Wpisz w pole cyfrę")

Zawartość błędu czyścimy przez Err.Clear.

### On Error Goto Miejsce

Jest to bardziej zaawansowana obsługa błędów, pozwala na prostsze reagowanie na błędy. Program po napotkaniu na błąd przenosi nas do odpowiedniego miejsca w kodzie,

wygląda to tak:

```
On Error Goto WystapilBlad:
```

```
Wynik = 6 / "C"
```

```
... dlasze dzialania
```

```
Exit Sub
```

```
WystapilBlad:
```

```
Msgbox "Wystapil blad nr " & Err.Number & ": " & Err.Description
```

Można w ten sposób obsłużyć parę błędów za jednym razem, użycie Exit Sub powoduje że program kończy procedure, inaczej msgbox zawsze byłby wyświetlany (nawet gdyby błąd nie istniał).

## 37. Schowek Windows

<http://visual.basic.kaem.forall.pl/>

Cóż, schowek Windows może zapamiętać 1 tekst i 1 obrazek (tak mi się wydaje i tak jest w vb). Schowek jest ogólny i przechowuje różne teksty i grafikę z innych programów. Słowo kluczowe to Clipboard, ma ono oczywiście różne atrybuty, o nich poniżej.

Gdy chcemy przesłać tekst do schowka piszemy :

```
Clipboard.SetText Zmienna
```

Teraz w schowku mamy już tekst w postaci naszej zmiennej, gdy chcemy natomiast wywołać ten tekst to :

```
Zmienna = Clipboard.GetText
```

Czyszczenie Schowka (Grafiki i tekstu) :

```
Clipboard.Clear
```

Grafika :

Dodawanie do schowka grafiki :

```
Clipboard.SetData = ZmiennaObrazek
```

Pobieranie grafiki ze schowka :

```
ZmiennaObrazek = Clipboard.GetData
```

Teraz niewielki przykład, na formie umieszczamy image1 i button. Klikamy dwukrotnie na button i piszemy :

```
Image1.Picture = Clipboard.GetData
```

Teraz uruchommy super edytor graficzny paint, narysujmy coś zaznaczmy i skopiujmy. Jeśli wszystko jest ok, to po kliknięciu w programie buttonu powinien w image1 pojawić się obrazek.

**Przykład:**

W przykładzie pokazane jest jak za pomocą timera przechwycić tekst ze schowka i

automatycznie użyć go w programie jeżeli spełnia nasze wymagania. Ustawmy na formie Timer o Interval = 500 i napiszmy dla niego zdarzenie:

```
Private Sub Timer1_Timer()  
If Len(Clipboard.GetText) > 6 Then  
    If Mid(Clipboard.GetText, 1, 7) = "http://" Then  
        MsgBox "w schowku jest link, teraz można go wykorzystać :)"  
        Timer1.Enabled = False  
    End If  
End If  
End Sub
```

W przykładzie jeżeli tekst w schowku ma długość większą niż 6 znaków i jeżeli pierwszymi znakami są 'http:/' to zachodzi zdarzenie (w tym wypadku tylko wyświetlenie komunikatu ale może być to na przykład łączenie z wybraną witryną internetową).

## 38. Klawisze

<http://visual.basic.kaem.forall.pl/>

Teraz opiszę jak obsługiwać klawisze, włączmy kod źródłowy jakiegoś okienka, z pierwszej listy (tej nad kodem) należy wybrać "Form" a z drugiej zdarzenie "KeyDown", teraz nasz program reaguje na wciskanie klawiszy. Zmienna przechowująca numer klawisza to zmienna KeyCode, żeby sprawdzić numery klawiszy możemy napisać :

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)  
a = MsgBox("Numer klawisza : " & KeyCode & ", Odpowiednik : " & Chr(KeyCode))  
End Sub
```

Włączmy program i wcisnijmy dowolny klawisz. Przykładowe klawisze to np. : 13 - Enter, 27 - Escape, 8 - Backspace. Teraz dodajmy na formę label1, sprawdźmy numery klawiszy dla strzałki w lewo (powinno być 37) i w prawo (a tu 39), następnie zmienimy procedurę na :

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)  
If (KeyCode = 37) Then Label1.Left = Label1.Left - 100  
' Jeżeli klawisz to strzałka w lewo to przesun label1 o 100 w lewo  
If (KeyCode = 39) Then Label1.Left = Label1.Left + 100  
' Jeżeli klawisz to strzałka w prawo to przesun label1 o 100 w prawo  
End Sub
```

Powinniśmy teraz strzałkami w lewo i prawo sterować label1. Funkcja ta nie działa jednak gdy na formie mamy obiekt z focusiem (focus to aktywność (obiekty te to np.button, text itp)). Należy wtedy dodać zdarzenia KeyDown tym obiektom, zmusza to jednak do dodawania naszej procedury do każdego obiektu. Dodajmy jeden command i zmienimy kod na :

```
Public Sub Klawisz(KeyCode As Integer)  
' Nasza funkcja która sprawdza klawisze  
If (KeyCode = 37) Then Label1.Left = Label1.Left - 100  
If (KeyCode = 39) Then Label1.Left = Label1.Left + 100  
End Sub
```

```

Private Sub Command1_KeyDown(KeyCode As Integer, Shift As Integer)
Klawisz (KeyCode)
' Jeżeli wciśniesz klawisz to przekaż go do funkcji Klawisz()
End Sub
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
Klawisz (KeyCode)
' Jeżeli wciśniesz klawisz to przekaż go do funkcji Klawisz()
End Sub

```

Teraz program reaguje na klawisze nawet jak ma obiekt z fokusem, rozwiązanie to ma jednak jedną wadę, mianowicie nie reaguje na klawisze zdarzeń (enter itp), gdyż te klawisze są przypisane obiektom z fokusem. Aby działały klawisze enter musimy niestety nie wstawiać tych obiektów.

Zamiast porównywać klawisze do ich numerów to możemy je porównać je do kodu klawiszy z visual basic-a :

do numeru :

```
if(KeyCode=13) Then...
```

do kodu vb :

```
If(KeyCode=vbKeyReturn) Then...
```

Pozwala to na łatwiejsze zapamiętanie klawiszy i nie trzeba ich sprawdzać za każdym razem :

Tabela kodu klawiszy vb :

vbKeyA - klawisz A

vbKeyB - klawisz B

...

vbKeyZ - klawisz Z

vbKey0 - klawisz 0

vbKey1 - klawisz 1

...

vbKey9 - klawisz 9

vbKeyF1 - klawisz F1

vbKeyF2 - klawisz F2

...

vbKeyF12 - klawisz F12

reszta klawiszy to ich angielska nazwa :

vbKey + nazwa, przykłady :

vbKeyUp - strzałka w górę

vbKeyDelete - klawisz Delete

vbKeySpace - spacja

Możemy także sprawdzać czy są wciśnięte kombinacje klawiszy, należy stosować do tego specjalne maski (dla shifta, ctrl'a i al'ta), przeanalizuj skrypt (na formie ma być tylko label1) :

```

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
ShiftDown = (Shift And vbShiftMask) > 0
' Definiujemy maske shifta
CtrlDown = (Shift And vbCtrlMask) > 0
' ctrl'a
AltDown = (Shift And vbAltMask) > 0

```



```

' alta
Maski = ""
If (ShiftDown = True) Then Maski = "SHIFT + "
' jeżeli shift jest trzymany to dodaj to do maski ("Maska" jest tutaj zwykłą zmienną
tekstową)
If (CtrlDown = True) Then Maski = Maski & "CTRL + "
' sprawdź ctrl'a i go dodaj gdy wciśnięty
If (AltDown = True) Then Maski = Maski & "ALT + "
' sprawdź altaa i go dodaj gdy wciśnięty
Label1.Caption = Maski
End Sub

```

## 39. Shell

<http://visual.basic.kaem.forall.pl/>

Funkcja shell pozwala nam uruchomić dowolny program (\*.exe) z poziomu naszego programu. Oprócz plików exe istnieje możliwość wykonywania plików z innymi rozszerzeniami za pomocą skojarzonych z nimi aplikacji.

Sposób wywołania funkcji shell jest następujący:

**Shell(ścieżka\_do\_programu, typ\_okna)**

Gdzie:

Ścieżka\_do\_programu to..... ścieżka do uruchamianego programu

Typ\_okna to parametr nieobowiązkowy określający rodzaj okna, w którym zostanie program uruchomiony

Dostępne typy:

Litera	Wartość	Opis
VbHide	0	Okno jest ukryte i ma focus
VbNormalFocus	1	Okno ma normalny rozmiar i posiada focus
VbMinimizedFocus	2	Okno jest wyświetlane jako ikona w pasku Start i ma focus
VbMaximizedFocus	3	Okno jest zamaksymalizowane i posiada focus
VbNormalNoFocus	4	Okno ma normalną wielkość i bieżące okno posiada focus
VbMinimizedNoFocus	6	Okno jest zminimalizowane i bieżące okno posiada focus

Przykład wywołania Kalkulatora w zminimalizowanym oknie bez focusu:

**Shell ("C:\Windows\calc.exe",6)**

Shell oferuje też dla użytkowników systemów serii Win9x przyjazny programik o nazwie Start. Za jego pomocą możemy nie tylko uruchamiać pliki exe, ale też:

- dowolny plik skojarzony z jakąś aplikacją, np.:

**Shell "Plik.exe C:\readme.txt"**

Spowoduje otwarcie pliku readme.txt za pomocą powiedzmy notatnika (programu skojarzonego z Plik .exe)

## 40. Rejestr

<http://visual.basic.kaem.forall.pl/>

Chyba każdy wie, co to jest rejestr. Więc nie ma sensu tego tłumaczyć. Standardowo Visual Basic pozwala na operacje tylko w obrębie klucza HKEY\_CURRENT\_USER\Software\VB and VBA Program Settings. Nie można schodzić poniżej tego klucza, chyba że używamy funkcji API. Opisane tu funkcje nie powinny zaszkodzić naszemu rejestrowi, w przypadku jakiegś awarii:

**DeleteSettings** - usuwa klucz wraz z jego zawartością  
**SaveSettings** - aktualizuje lub dodaje klucz  
**GetAllSettings** - zwraca listę kluczy i ich wartości  
**GetSettings** - zwraca wartość klucza

Oto schemat klucza tworzonego przez SaveSettings:

HKEY\_CURRENT\_USER\Software\VB and VBA Program  
Settings\klucz\_aplikacji\klucz\_sesji\klucz

Tworzenie klucza:

`SaveSetting "klucz_aplikacji", "klucz_sesji", "klucz", "wartość"`

Pobieranie wartości klucza:

`GetSetting "klucz_aplikacji", "klucz_sesji", "klucz", "wart_dom"`

Jego wywołanie nie różni się prawie od poprzedniego, oprócz pojawienia się wartości domyślnej wart\_dom, zwracanej, gdy klucz nie istnieje. Np.:

`a = GetSetting "vircom", "program", "imie", "nie_ma"`

Spowoduje to zwrócenie wartości Vircom, ale gdyby klucz nie istniał to "nie\_ma"

**GetAllSettings**

Pobiera z klucza podrzędnego, wszystkie klucze i ich wartości. Wywołanie ma postać:

`a = GetAllSettings "klucz_aplikacji", "klucz_sesji" I przykład:`

```
Text1.text = ""
Dim Odczytane As Variant
Dim x As Integer
Odczytane = GetAllSettings("Kalendarz", "Dziadek")
For x = 0 To UBound(Odczytane)
Text1.Text = Text1.Text & Odczytane(x, 0) _
& vbTab & " =" & Odczytane(x, 1) & vbNewLine
Next x
```

**DeleteSetting**

Usuwa klucz i ewentualnie jego podwartości. Wywołanie ma postać:

DeleteSetting "klucz\_aplikacji", "klucz\_sesji", "klucz"

Do rejestru możemy zapisywać ważne informacje typu czy użytkownik jest zarejestrowany, ilość włączeń programu itp.

## 41. Menu

<http://visual.basic.kaem.forall.pl/>

W tym dziale opisze w jaki sposób stworzyć menu typu Plik > Otwórz itp.. Cały proces jest dosyć prosty. Klikamy raz na formę, a potem z górnego menu (takiego jak właśnie robimy) wybieramy opcję Tools > Menu Editor.... Ukazuje się nam okienko edycji. W pole Caption wpisujemy jakąś nazwę (w przykładzie "Plik") a w miejscu Name jakąś nazwę. Po kliknięciu ok. na naszym programie będzie już coś podobnego do menu. Aby dodać kolejny Caption klikamy na insert i znowu wypełniamy pola Caption i Name (Name musi być inne od poprzedniego Name chyba że ustawimy index-y, ale o indexach troszke później).

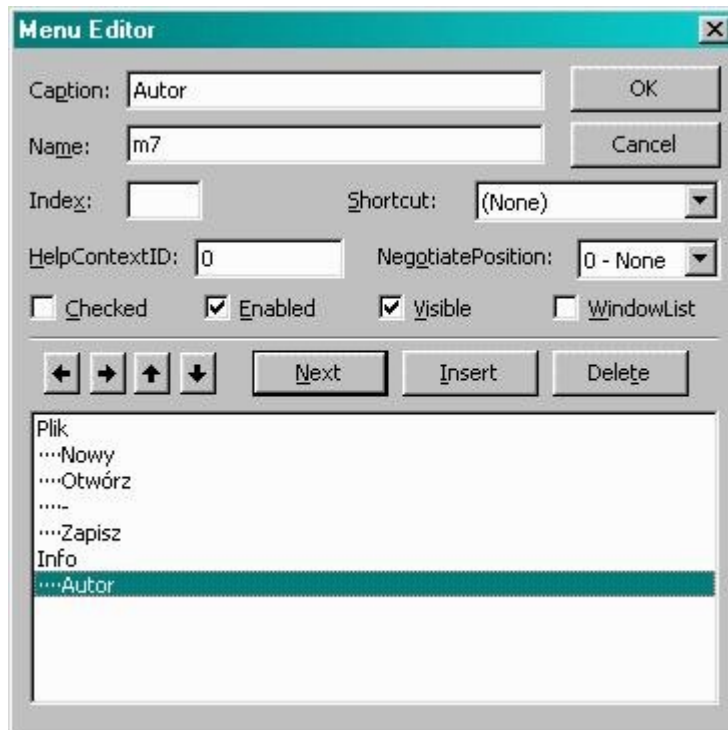


Chcąc dodać podmenu musimy kliknąć nastrzałkę w prawo, wtedy tworzymy podmenu, możemy także stworzyć podmenu dla podmenu, aby to lepiej zajarzyć spójrzcie na poniższy rysunek. Możemy także tworzyć paski odstępu robimy to wpisując w Caption -. To także pokazane jest na obrazku. Teraz aby zobaczyć czy nasze menu działa kliknijmy ok., jeśli wyskoczy jakiś błąd to najprawdopodobniej nie wypełniliśmy któregoś pola Name. Możemy dodać do naszych opcji klawisz skrótu, robimy to wybierając z listy Shortcut odpowiednią kombinację klawiszy. Aby stworzyć procedurę dla każdego przycisku z menu należy po prostu w zamknąć menu editor i kliknąć na wybraną pozycję. Dla opcji z menu o Name m1 powinno wyglądać to tak :

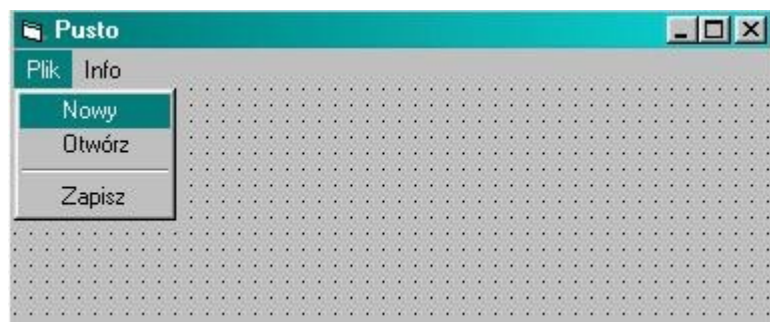
Private Sub m1\_Click()

End Sub

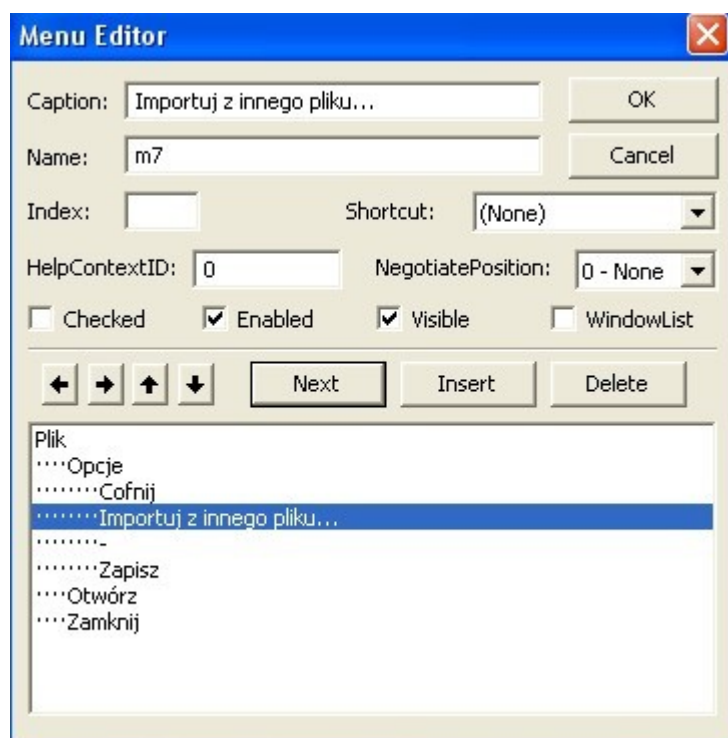
Pomiędzy tym wstawiamy nasze polecenia.



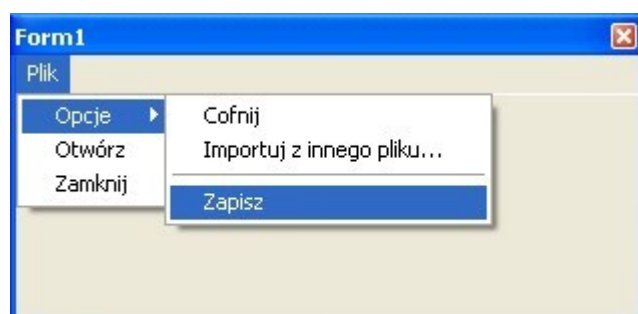
Gotowe menu powinno wyglądać następująco:



Może oczywiście tworzyć zagnieżdżone w sobie elementy (dowolną ilość razy)



Co da nam:



## 42. Pasek statusu

<http://visual.basic.kaem.forall.pl/>

Aby dodać do swojego programu pasek statusu (taki pasek od dołu programu) należy najpierw dodać kontrolki "Microsoft Windows Common Controls 6.0", wybieramy je oczywiście z listy komponentów (CTRL + T). Wśród nowo dodanych komponentów znajdujemy StatusBar i umieszczamy go na formie. Teraz aby pasek lepiej wyglądał to zmieniamy mu właściwość Style na 1 - sbrSimple. Powinniśmy uzyskać efekt podobny do:



czegoś takiego. Właściwość odpowiedzialna za tekst to SimpleText. Zrobmy więc prościutki przykładzik z 2 przyciskami. Po kliknięciu jednego :

```
Private Sub Command1_Click()  
    StatusBar1.SimpleText = "Sprawdzanie danych..."  
End Sub
```

```
Private Sub Command2_Click()  
    StatusBar1.SimpleText = "Błąd programu !!!"  
End Sub
```

Jest to najczęściej wykorzystywane użycie tego paska.

## 43. Menu graficzne

<http://visual.basic.kaem.forall.pl/>

Aby zrobić w naszym programie graficznym musimy przede wszystkim dodać zestaw kontrolki : "Microsoft Windows Common Controls 6.0" (CTRL + T). Teraz na naszej formie umieszczamy Toolbar i ImageList, klikamy na umieszczoną kontrolkę ImageList prawym przyciskiem myszy i wybieramy Properties. Włącza nam się nowe okno, najpierw wybieramy rozmiar obrazka, najlepiej 16 x 16. W następnej zakładce "Images" dodajemy nasze ikonki do menu klawiszem Insert Picture. Dodajmy na razie tylko z 3 obrazki. Należy zauważyć, że każdy obrazek ma swój Index, kolejno jest to 1, 2, 3, 4, 5.....

Teraz klikamy prawym przyciskiem na Toolbar1 i znów wybieramy Properties. Po umieszczeniu na formie nasz Toolbar powinien być przy samej górze. We właściwości General znajdujemy trzecią opcję "ImageList" i z rozwijanego menu wybieramy naszą listę obrazków. Przechodzimy do drugiej zakładki (Buttons) i aby dodać przycisk klikamy na InsertButton. Szare pola powinny nam się aktywować zmieniając kolor na biały. W miejsce Caption wpisujemy tekst do obrazka, w miejscu Key wpisujemy nazwę naszego przycisku, będzie ona potrzebna gdy będziemy chcieli stworzyć procedurę po kliknięciu na nasz przycisk. ToolTipText to oczywiście tekst pojawiający się po najechaniu. Pole image wyznacza nam który obrazek ma być wyświetlony. Jeśli więc nasz obrazek ma Index = 1 to wpisujemy tutaj liczbę 1. Chcąc dodać kolejny obrazek znów klikamy Insert Button i wypełniamy pola. Klikamy ok i po uruchomieniu programu naszym oczom powinien pokazać się podobny widok do tego poniżej.



Możemy także pobawić się wyglądem paska (Appearance, BorderStyle, Style, TextAlignment i Align)



Aby uzyskać różne efekty zmieniamy Styl przycisku (Prawym na Toolbar, Properties i zakładka Button) na :

- 1 - zwykły przycisk
- 3 - przedziałka
- 5 - strzałka rozwijania

Inne opcje stylu w tym przypadku są zbędne. Upewnijmy się że ustawiliśmy jakieś nazwy w opcji Key dla każdego przycisku, ja ustawiłem kolejno p1, p2, p3, p4. Aby utworzyć procedurę do tego menu musimy najpierw dwukrotnie kliknąć na nasz Toolbar.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
End Sub
```

Sama procedura nie rozpoznaje jednak który z przycisków został kliknięty, musimy użyć tu opcji Key :

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
MsgBox (Button.Key)
End Sub
```

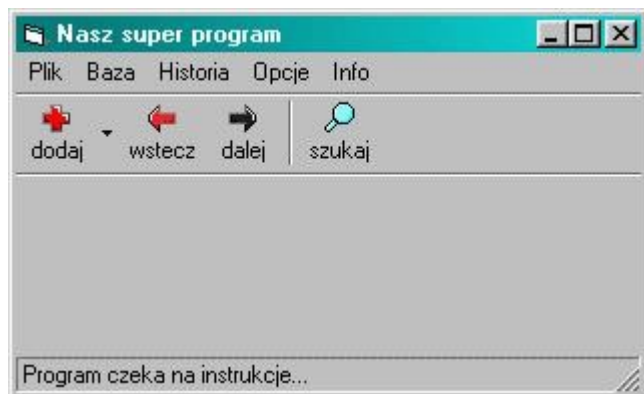
Możemy też w łatwy sposób posłużyć się funkcja Case :

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
Select Case Button.Key
Case Is = "p1":
MsgBox "procedury dla przycisku 1"
Case Is = "p2":
MsgBox "procedury dla przycisku 2"
End Select
End Sub
```

Do każdego przycisku możemy dodać rozwijaną listę opcji. Najpierw ustawiamy

przyciskowi Style na 5, a potem możemy dowolnie dodawać opcje klikając na Insert ButtonMenu. Dla każdego ButtonMenu musimy także zadeklarować pole Key.

Łącząc wszystkie poznane menu i pasek statusu możemy stworzyć program z ładnym i intuicyjnym interfejsem.



## 44. Czas

<http://visual.basic.kaem.forall.pl/>

Aby uzyskać teraźniejszy czas (dokładnie ten z systemu) musimy posłużyć się funkcją Now(). Robimy to następująco:

### Msgbox Now()

Dwie pierwsze liczby to rok, następne to miesiąc, dzień, godzina, minuta, sekunda.

Do wybierania konkretnych wartości z daty służą polecenia

Year(Data) - zwraca rok z konkretnej daty

Month(Data) - miesiąc

Day(Data) - dzień

Hour(Data) - godzina

Minute(Data) - minuta

Second(Data) - sekunda

Wyświetlenie godziny z bieżącej daty

### Msgbox "Jest godzina: " & Hour(Now())

Zamiast funkcji Now() Możemy użyć funkcji wyświetlających sam rok/miesiąc/dzień lub samą godzinę/minutę/sekundę. Posłużą nam do tego funkcje Date i Time.

### Msgbox Date

### Msgbox Time

Dostaniemy dwa komunikaty w pierwszym rok/mi..... a w drugim godzinę/se.....

### DateSerial i TimeSerial

Dwie powyższe funkcje zamieniają czas podany przez nas na czas w postaci systemowej, przydają się gdy chcemy porównywać jakieś czasy ze sobą.



`czas = DateSerial(rok,miesiac,dzień)`

Podając za rok 2003, miesiąc 5 a dzień 12 otrzymamy wynik w postaci 03-05-12.

`czas = TimeSerial(godzina,minuta,sekunda)`

Wpisując odpowiednio 15, 58, 12 otrzymamy 15:58:12

### **DateDiff**

Bardzo przydatna funkcja, która zwraca ilość jednostek czasu pomiędzy dwoma datami.

`ile = DateDiff(co,czas1,czas2)`

Dokładnie to opisując to obliczona zostaje ilość co (patrz tabela) od daty pierwszej do daty drugiej. Tabela co :

yyyy - rok  
q - kwartał  
m - miesiąc  
d - dni  
w - weekendy  
ww - tygodnie  
h - godziny  
n - minuty  
s - sekundy

Dla przykładu sprawdźmy ile dni zostało do roku 2010 :

```
Private Sub Command1_Click()  
    czas1 = Date  
    czas2 = DateSerial(2010, 1, 1)  
    ile = DateDiff("d", czas1, czas2)  
    MsgBox ile  
End Sub
```

Możemy uzyskać także liczbę godzin, minut itp., wstawiając odpowiednio zamiast "d" jakieś inne znaki z tabelki. DateDiff przydaje nam się gdy chcemy zrobić demo naszego programu działające do określonej daty. Gdy czas1 jest większy od czas2 to ile przyjmuje wartość ujemną możemy więc napisać

```
If ile<0 Then Unload Form1  
DateAdd  
DateAdd(co,iletego,czas)
```

Dodaje co (patrz tabela z yyyy,q,m,...) w liczbie ile tego do czas. Chcąc dodać 3 miesiące do dzisiaj napiszemy :

`nowoczes = DateAdd("m",3,Date)`

Dowiemy się jaka data będzie za 3 miesiące.

### **DatePart**

Zwraca ilość jednostek które minęły od początku roku

```
DatePart(co,czas)
```

Obliczy nam ile co minęło w roku podanym w czasie np:

```
ile = DatePart("d",Date)
```

Zmienna ile będzie przechowywała ilość dni od początku bieżącego roku.

### Timer

Timer liczy nam czas w sekundach jaki minął od godziny 0:00. Możemy w ten sposób obliczyć czas wykonywania funkcji, pętli itp. :

```
czas1 = Timer  
czas2 = Timer  
roznica = czas2 - czas1
```

Oblicza czas między czas1 a czas2. Jest to przydatne dla zaawansowanych funkcji lub gdy przyspieszamy program i chcemy sprawdzić czy go naprawdę przyspieszyliśmy, a nie zwolniliśmy.

## 45. Indeksowanie obiektów

<http://visual.basic.kaem.forall.pl/>

Aby ułatwić nieco działanie programu możliwe jest indeksowanie obiektów. Zaczniemy indeksować na przykładzie pięciu pól tekstowych. Najpierw kładziemy jedno pole tekstowe i dajemy mu Name = Pole, zmieniamy też właściwość Index na 0. Teraz kopiujemy nasze pole i wklejamy 4 razy. Mamy już na formie 5 zindeksowanych pól Tekstowych. Ustaw na formie przycisk i spróbuj wpisać po kliknięciu, żeby wyświetliło tekst naszego pola tekstowego.

```
Private Sub Command1_Click()  
MsgBox Pole.Text  
End Sub
```

I co błąd, no tak. Dodając indeks nasze pola przyjmują liczby w nawiasie, wygląda to więc tak :

```
Private Sub Command1_Click()  
MsgBox Pole(0).Text  
End Sub
```

Zastosowanie : Możemy teraz łatwo zapętlić program i wydobyć właściwości ze wszystkich pól na raz :

```
Private Sub Command1_Click()  
Dim a  
Do While(a<5)  
MsgBox Pole(a).Text  
a=a+1  
Loop
```

End Sub

Normalnie musielibyśmy napisać tak :

```
Private Sub Command1_Click()  
MsgBox Pole(0).Text  
MsgBox Pole(1).Text  
MsgBox Pole(2).Text  
MsgBox Pole(3).Text  
MsgBox Pole(4).Text  
End Sub
```

Przy większej ilości elementów to bardzo dobre udogodnienie. Kliknijmy teraz dwukrotnie w nasz Pole.

```
Private Sub Pole_Change(Index As Integer)
```

End Sub

Doszła nam zmienna Index która jest numerem pola tekstowego, teraz jedna procedura zastępuje nam wszystkie. Np. :

```
Private Sub Pole_Click(Index As Integer)  
MsgBox "Klikłeś w : " & Index  
End Sub
```

Kolejnym zaletom indeksowania obiektów jest możliwość łatwego tworzenia ich sobowtórów o nowym indeksie. Umieśćmy na formie jeden przycisk (Command1) i jedno pole tekstowe (Pole) z indeksem ustawionym na 0. Teraz po kliknięciu w przycisk napiszmy :

```
Public a As Integer  
Private Sub Form_Load()  
a = 1  
End Sub  
Private Sub Command1_Click()  
Load Pole(a)  
Pole(a).Left = Pole(a - 1).Left  
Pole(a).Top = Pole(a - 1).Top + 300  
Pole(a).Visible = True  
a = a + 1  
End Sub
```

Teraz po kliknięciu ładuje się nowe pole o indeksie a. Następnie ustalwamy jego pozycje (Left i Top) i wyświetlamy je (Visible = True). Teraz już tylko zwiększamy a o 1 żeby nie było dwóch takich samych obiektów o tym samym indeksie. Uruchommy program i kliknijmy pare razy na przycisk.

Poleceniem Unload Pole(a) usuwamy wybrany obiekt o indeksie a.

## 46. Przeciąganie obiektów

<http://visual.basic.kaem.forall.pl/>

Jest to jedna z często używanych opcji windows. Aby ją zilustrować opisze to najpierw na 2 polach TextBox. Umieścimy na formie dwa TextBox-y, aby możliwe było przeciąganie musimy ustawić im opcje DragMode na 0, wypadło by także ustawić DragIcon (ikonę przeciągania) na jakąś inną niż zwykła strzałka kursora. Inaczej użytkownik nie wiedziałby że przeciąga jakiś obiekt. Ustawmy także Texty na "Visual" i "Basic". Po uruchomieniu programu kliknijmy lewym przyciskiem myszki na polu tekstowym i trzymając go przeciągnijmy w inne miejsce, kursor powinien zmienić się na nasz wybrany. Teraz wypadło by ustawić procedury do przeciąganie obiektów. Najsamprzód ustawmy dla Text1 proceduręMouseDown i wpiszmy w niej :

**Text1.Drag 1**

Informuje to program że zaczęto przeciąganie, a przeciągany obiekt to nasz Text1.

Teraz musimy ustawić procedure dla Text2 aby reagowało na postawienie na nim obiektu, wybieramy więc dla Text2 procedure DragDrop (upuszczenie obiektu) i wpisujemy :

**If Source = Text1 Then Text2.Text = Text1.Text**

Jeżeli przeciągamy Text1 to Text2.Text ma być równy Text1.Text

Teraz po przeciągnięciu Text1, Text2 dostaje wartość Text1. Dla treningu spróbuj ustawić to tak aby po przeciągnięciu Text2 na Text1, Text1 dostawał wartość Text2. Ciekawszym wykorzystaniem tej funkcji jest użycie jej dla obiektów takich jak ListBox (Należy pamiętać o ustawieniu DragMode na 0, dodajmy także trochę item do List1, aby było co przeciągać). W List1\_MouseDown piszemy :

**List1.Drag 1**

**A w List2\_DragDrop :**

```
If Source = List1 Then
List2.AddItem (List1.Text)
List1.RemoveItem List1.ListIndex
End If
```

Program powinien po upuszczeniu obiektu z List1 na List2 dodawać do List2 zaznaczony List1.Text, a potem kasować z List1 ów właśnie przenoszony tekst. Co jednak jeśli List1 jest puste, a my nadal przeciągamy ? Wskoczy błąd, zapobiegamy mu stosując sprawdzenie czy List1 ma jeszcze obiekty (List1\_MouseDown) :

**If List1.ListCount > 0 Then List1.Drag 1**

Teraz możemy z łatwością przeciągać obiekty z List1 do List2. Zrobmy także procedure odwrotną czyli przeciąganie obiektów z List2 do List1. Przeciągać możemy prawie wszystko, możemy także mieszać przeciągane elementy i przekazywać różne wartości nie tylko tekst.

Dzięki przeciąganiu możemy także poruszać obiektami na formie, wystarczy dla danego obiektu zrobić obsługę przeciągania, a na formie lub innym obiekcie z uchwytem po którym chcemy przeciągać obiekt procedure upuszczania. Wyglądało by to mniej więcej

tak:

```
Private Sub Text1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Text1.Drag
End Sub
```

```
Private Sub Form_DragDrop(Source As Control, X As Single, Y As Single)
Text1.Top = Y
Text1.Left = X
End Sub
```

Opisując te dwie procedury: pierwsza ustawia przeciąganie dla pola Text1 gdy klawisz myszy jest wciśnięty, druga procedura ustawia nowe położenie X i Y dla pola Text1 gdy zostanie "położone" na Form1. Robi to jednak w pozycji kursora nie uwzględniając miejsca w którym "trzymamy" obiekt. Należało by zastosować kilka poprawek:

```
Private Lewo As Integer
Private Gora As Integer
```

```
Private Sub Text1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Lewo = X
Gora = Y
Text1.Drag
End Sub
```

```
Private Sub Form_DragDrop(Source As Control, X As Single, Y As Single)
If Source = Text1 Then
Text1.Left = X - Lewo
Text1.Top = Y - Gora
End If
End Sub
```

Użyte zostały dwie zmienne Lewo i Góra które przechowują położenie kursora w stosunku do naszego obiektu, wystarczy je potem odjąć od nowych X i Y, w drugiej procedurze także została wprowadzona poprawka, sprawdzane jest czy naszym przeciąganym obiektem napewno jest Text1.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 47. Forma – rysowanie

<http://visual.basic.kaem.forall.pl/>

Na naszym formularzu możemy rysować stawiając piksele, a także pisać tekst. Aby postawić jakiś piksel wystarczy napisać :

PSet (X,Y)

Najlepszy efekt uzyskamy zmieniając we właściwościach formy ScaleMode na 3 - pixel. Należy też pamiętać, że nie można rysować w zdarzeniu Form\_Load

```
Private Sub Form_Load()  
PSet (10, 10)  
End Sub
```

Powyższy kod niestety nie wygeneruje nam piksela, musimy pisać to więc np. w Timerze lub po kliknięciu w przycisk :

```
Private Sub Command1_Click()  
PSet (10, 10)  
End Sub
```

Kolor piksela zmieniamy funkcją ForeColor :

```
ForeColor = RGB(255, 0, 0)
```

Natomiast wielkość funkcją :

```
DrawWidth = 10
```

Możemy też rysować linie :

```
Line (X1, Y1)-(X2, Y2)
```

I koła :

```
Circle (X1, Y1), Promień
```

Dla przykładu pętla malująca gradient na formie

```
Private Sub Command1_Click()  
a = 0  
Do While (a < Form1.Height) ' powtarzaj dopóki a jest mniejsze od wysokości formy  
ForeColor = RGB(a, a, a) ' ustaw nowy kolor  
Line (0, a)-(Form1.Width, a) ' narysuj poziomą linię na szerokość formy  
a = a + 1 ' zwiększ a  
Loop  
End Sub
```

## 48. Forma – autoredraw

<http://visual.basic.kaem.forall.pl/>

Właściwość autoredraw decyduje o zapamiętywaniu obrazka, ustawiona na true każe zapamiętać obrazek programowi w pamięci a ustawiona na false zapamiętuje go tylko czasowo, dla przykładu umieścimy na formie dwa obiekty picture (z jakimiś obrazkami), jednemu nadajmy true a drugiemu false. Po uruchomieniu programu zminimalizujemy go do paska i przywróćmy. Jak zauważyłeś tam gdzie autoredraw = true obrazek pozostał.

Teraz zrobimy to samo tyle że nasze 2 picture muszą być puste. Napišmy procedure, która po przycisku rysuje w każdym z pól parę pikseli :

```
Picture1.Pset(10,10)  
Picture2.Pset(10,10)
```

Musimy oczywiście pamiętać o przemienieniu scalemode z 1-twip na 3-pixel. Po uruchomieniu programu i kliknięciu w przycisk który rysuje piksele zauważamy że piksele pokazały się tylko na obiekcie z autoredraw = false. Tak naprawdę na drugim obiekcie także pokazał się pixel, tyle że trzeba obiekt odświeżyć.

`Picture2.Refresh`

Autoredraw jest o tyle ważne, bo gdy tworzymy jakiś program z grafiką, a ktoś daje do paska i potem traci całą grafikę to raczej wyłączy on nasz wadliwy program. Pozostaje więc na do wyboru

False - nie trzeba odświeżać, ale obrazek znika po minimalizowaniu i zakryciu innym oknem

True - trzeba odświeżać, ale obrazek zawsze pozostaje

## 49. Forma – pisanie

<http://visual.basic.kaem.forall.pl/>

Oprócz umieszczania na formie tekstu w postaci label możemy sami pisać tekst w wybranych miejscach. Służą do tego 3 funkcje :

`Form1.CurrentX = 10`

Ustawienie miejsca w poziomie

`Form1.CurrentY = 10`

Ustawienia miejsca w pionie

`Form1.Print "???"`

Wypisanie tekstu.

Oczywiście zamiast Form1 możemy użyć innej nazwy lub pola obrazka. Trzy powyższe funkcje napiszą zaczynając od miejsca 10x10 tekst ???. Zamiast tych trzech funkcji możemy sami napisać własną :

```
Public Sub Pisz(ByVal X As Integer, ByVal Y As Integer, ByVal Tekst As String)
Form1.CurrentX = X
Form1.CurrentY = Y
Form1.Print Tekst
End Sub
```

Teraz piszemy własną formułą :

`Pisz 10, 10, "???"`

Możemy także łatwo zmieniać atrybuty czcionki, kolor :

`Form1.ForeColor = RGB(0,100,200)`

Pogrubienie :

`Form1.FontBold = True`

Przekrzywienie :

`Form1.FontItalic = True`

Rozmiar :

`Form1.FontSize = 20`

Podkreślenie:

`Form1.FontUnderline = True`

## 50. O API

<http://visual.basic.kaem.forall.pl/>

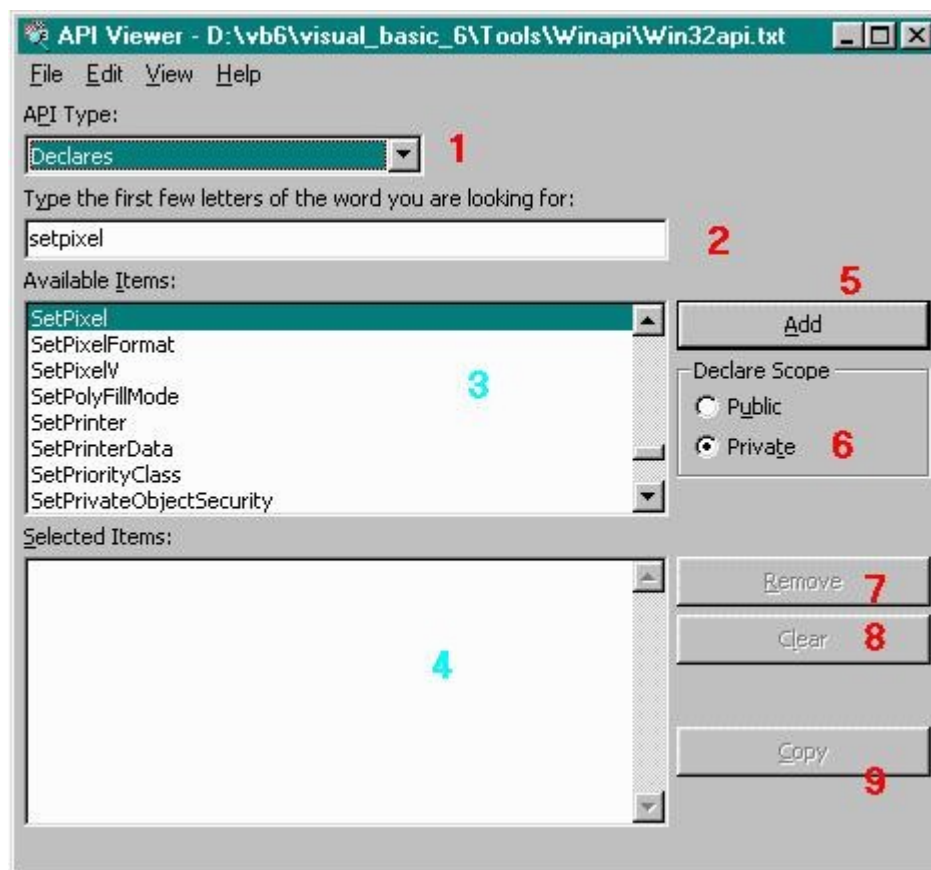
Api są to procedury windowsa, w VB można je łatwo wywołać i stosować.

Obsługa API-Text Viewera

API-Text Viewer służy nam do szybkiego i sprawnego wyszukiwania interesujących nas funkcji, procedur, czy stałych API. Odnajdzie także na nasze polecenie dowolny typ UDT.

Jego obsługa jest banalna i sprowadza się tylko do wypełniania odpowiednich pól. Uruchomić ten program możemy przez menu start lub przez Visual Basic z menu Diagram | API Viewer po wcześniejszym dodaniu w Add-in menager. Oto wygląd API-Viewera:





- 1-Wyberamy, co nam jest potrzebne: deklaracja funkcji, stała, czy typ
- 2-Tutaj wpisujemy nazwę na podstawie której nastąpi wyszukiwanie
- 3-Tutaj pojawia się odpowiedzi na szukanie
- 4-Po zatwierdzeniu wyszukanej funkcji i naciśnięciu przycisku Add, pojawi się kod do wklejenia do programu
- 5-Zatwierdzamy nasz wybór
- 6-Określamy zasięg naszej deklaracji
- 7-Usuwamy wybraną część funkcji z okienka 4
- 8-Czyści ekran Selected Items
- 9-Kopiuje do schowka Windows zawartość pola Selected Items

Zanim jeszcze zaczniemy naszą zabawę z API musimy zdefiniować bazę w której ma szukać nasz program. W tym celu wybieramy File | Load Text File i otwieramy Win32api.txt.

## 51. Przeciąganie obiektów

<http://visual.basic.kaem.forall.pl/>

Do przeciągania obiektów posłużą nam 2 funkcje API (można je łatwo znaleźć w APT-Text-Viewer zamiast uczyć się na pamięć) :

`Private Declare Sub ReleaseCapture Lib "user32" ()`

`Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Long, IPParam As Any) As Long`

Pamiętajmy aby druga funkcja była w całości wpisana w jednej linijce. Pierwsza opcja informuje o puszczeniu przycisku myszki, a druga umieszcza przeciągany obiekt w odpowiednim miejscu. Dla zobrazowania opcji posłużmy się przykładem. Najpierw umieścmy na formie jakiś obrazek (Picture1). Teraz ustawmy mu wartość przeciągania na manulaną (DragMode = 0). Następnie wstawiamy procedure MouseDown do naszego obiektu :

```
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
ReleaseCapture
SendMessage Picture1.hWnd, &HA1, 2, 0&
End If
End Sub
```

Teraz wyjaśnienie. Jeżeli wciśnięty klawisz myszki lewy to przeciągaj obiekt, jeżeli upuszczony to umieść obiekt Picture.hWnd w wybrane miejsce. W ten sposób możemy przeciągać tylko obiekty mające właściwość uchwytu hWnd. Większość obiektów posiada tą właściwość. Należy także usunąć z danego obiektu funkcje Click, gdyż nie jest ona wykonywana, należy treść przypisać do funkcji MouseDown. Przykład na commandzie :

```
Private Sub Command1_Click()
MsgBox "Klikłeś"
End Sub
Private Sub Command1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
ReleaseCapture
SendMessage Command1.hWnd, &HA1, 2, 0&
End If
End Sub
```

Według toku myślenia kod powinien wyświetlić komunikat, jednak nie robi tego, aby wszystko było poprawnie musimy to napisać tak :

```
Private Sub Command1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
ReleaseCapture
SendMessage Command1.hWnd, &HA1, 2, 0&
End If
MsgBox "Klikłeś"
End Sub
```

Można w ten sposób umilić użytkownikowi korzystanie z programu bo będzie mógł samodzielnie poprzestawiać niektóre obiekty na formie. Kolejnym przykładem zastosowania przeciągania jest przypisanie przeciągania dla formularza, robimy to w identyczny sposób.

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 Then
ReleaseCapture
SendMessage Me.hWnd, &HA1, 2, 0&
End If
End Sub
```

Oczywiście w każdym przypadku muszą być zadeklarowane funkcje api. Przykład przeciągania formularza przydaje się bardzo dla okien z właściwością `BorderStyle` ustawioną na 0, 3, 4 lub 5.

## 52. Własny kształt formularza

<http://visual.basic.kaem.forall.pl/>

Każdy chciałby upiększyć swój program, by wyróżniał się z tłumu, można zrobić własny kształt dla formy. Lekcja będzie trudna więc radzę uważać. Pierwsza funkcja to :

```
Private Declare Function CreateRectRgn Lib "gdi32" (ByVal x1 As Long, ByVal y1 As Long,
ByVal X2 As Long, ByVal Y2 As Long) As Long
```

Wycina ona prostokąt z formularza.

```
pole = CreateRectRgn(x1, y1, x2, y2)
```

Następna funkcja pozwala wyświetlić tylko wyciętą część, więc nasz prostokąt :

```
Private Declare Function SetWindowRgn Lib "user32" (ByVal hWnd As Long, ByVal hRgn
As Long, ByVal bRedraw As Boolean) As Long
```

Wyglądało by to mniej więcej tak :

```
pole = CreateRectRgn(0, 0, 200, 200)
co = SetWindowRgn(Me.hWnd, pole, True)
```

Co nam jednak po zwykłym prostokącie ? Otóż oprócz prostokąta możemy wycinać Elipsy:

```
Private Declare Function CreateEllipticRgn Lib "gdi32" (ByVal x1 As Long, ByVal y1 As
Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
```

```
pole = CreateEllipticRgn(x1, y1, x2, y2)
```

Prostokąty z zaokrąglonymi rogami :

```
Private Declare Function CreateRoundRectRgn Lib "gdi32" (ByVal x1 As Long, ByVal y1
As Long, ByVal X2 As Long, ByVal Y2 As Long, ByVal X3 As Long, ByVal Y3 As Long) As
Long
```

```
pole = CreateRoundRectRgn(x1, y1, x2, y2, x3, y3)
```

Gdzie x3 i y3 to szerokość miejsca zaokrąglonego.

Na tym jednak nie koniec opcji wycinania i wyświetlania części formularzy. Otóż wycięte elementy można ze sobą łączyć i wyświetlać dzięki temu niepowtarzalne formy, służy do tego :

```
Public Declare Function CombineRgn Lib "gdi32" (ByVal hDestRgn As Long, ByVal
hSrcRgn1 As Long, ByVal hSrcRgn2 As Long, ByVal nCombineMode As Long) As Long
```

```
obszar = CombineRgn(pole, pole1, pole2, 2)
```

Powyższy przykład łączy pole1 i pole2 w nowe pole. Zamiast ostatniej wartości 2 możemy stosować także :

- 1 - wyświetla tylko część wspólną pola1 i pola2
- 2 - wyświetla połączone części pola1 i pola2
- 3 - wyświetla pole1 i pole2 bez ich części wspólnej
- 4 - wyświetla tylko część pola1 bez części wspólnej i bez pola2
- 5 - wyświetla kopie pola1

Poniżej przykładzik wyświetlający ciekawie wyglądającą formę :

```
pole5 = CreateRectRgn(0, 0, 200, 200)
pole6 = CreateRectRgn(0, 0, 200, 200)
pole7 = CreateRectRgn(0, 0, 200, 200)
pole1 = CreateEllipticRgn(0, 0, 150, 150)
pole2 = CreateEllipticRgn(0, 50, 150, 200)
pole3 = CreateEllipticRgn(50, 0, 200, 150)
pole4 = CreateEllipticRgn(50, 50, 200, 200)
CombineRgn pole5, pole1, pole2, 3
CombineRgn pole6, pole3, pole4, 3
CombineRgn pole7, pole5, pole6, 3
SetWindowRgn Me.hWnd, pole7, True
```

Uwaga pole5, 6 i 7 muszą być wcześniej zadeklarowane, inaczej funkcja CombineRgn nie połączyła naszych pól od 1 do 4. Teraz opis powyższego kodu.

1. Deklarujemy kwadrat 200x200 pikseli
2. Deklarujemy kwadrat 200x200 pikseli
3. Kolejny kwadrat 200x200
4. Wycinamy koło 150x150
5. Następne
6. Kolejne
7. No i ostatnie koło
8. Wycinamy z dwóch pierwszych kółek wszystko oprócz ich części wspólnej i przypisujemy to do pola 5
9. To samo robimy z kołkiem 3 i 4 tyle że przypisujemy wycięty obszar do pola6.
10. Wycinamy wszystko oprócz części wspólnej z pola5 i pola6 i przypisujemy to polu7.
11. Wyświetlamy naszą powycinaną formę (pole7).

Ciekawy efekt uzyskamy łącząc własne kształty formy z jej przeciąganiem. Należy z tym trochę poeksperymentować.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 53. GetPixel i SetPixel

<http://visual.basic.kaem.forall.pl/>

Funkcje te służą do stawiania i pobierania pikseli z formy lub obiektów type picture etc. Po dodaniu ich z ApiViewer powinniśmy dostać coś takiego :

```
Public Declare Function SetPixel Lib "gdi32" (ByVal hDC As Long, ByVal X As Long, ByVal Y As Long, ByVal crColor As Long) As Long
Public Declare Function GetPixel Lib "gdi32" (ByVal hDC As Long, ByVal X As Long, ByVal Y As Long) As Long
```

Oczywiście jeżeli funkcje dodajemy nie w module, a w form musimy zmienić Public na Private. Teraz chcąc ustawić piksel na ekranie piszemy :

```
SetPixel Form1.hDC, 10, 10, RGB(r, g, b)
```

Form1.hDC to nasz uchwyt, piksel możemy stawiać i pobierać z różnych uchwytów, np. z uchwytu picture (Picture1.hDC). Funkcja SetPixel jest o wiele szybsza od Pset. Pobieranie piksela jest procesem trochę trudniejszym gdyż należy jeszcze pobrany piksel rozłożyć na RGB :

```
kolor = GetPixel(Form1.hDC, 10, 10)
r = kolor Mod 256
b = Int(kolor / 65536)
g = (kolor - (b * 65536) - r) / 256
```

Teraz wiemy jaki piksel stoi w pozycji 10x10. Dla zapamiętania spróbuj zrobić pętlę która rysuje kwadrat (wypełniony), wykorzystaj GetPixel i stwórz jakiś ciekawy efekt.

## 54. BitBlt i StretchBlt

<http://visual.basic.kaem.forall.pl/>

BitBlt i StretchBlt to funkcje kopiujące grafike, są one bardzo przydatne gdyż szybko potrafią skopiować nawet duży obszar graficzny. Gdybyśmy chcieli skopiować grafike o wymiarach 100x100 metodą GetPixel zajęło by to mnóstwo czasu, a tutaj 0,0 s. Obydwie procedury dodajemy oczywiście z API-Viewera.

```
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
Private Declare Function StretchBlt Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal nSrcWidth As Long, ByVal nSrcHeight As Long, ByVal dwRop As Long) As Long
```

Wyglądają może trochę skomplikowanie, ale są bardzo proste w użyciu. Pierwsza kopiuje wybrany obszar w dane miejsce :

```
BitBlt Picture2.hdc, 0, 0, 200, 200, Picture1.hdc, 0, 0, &HCC0020
```

W ten sposób do Picture2 wkleimy obszar 200x200 z Picture1 wycinając w miejscu 0;0. Procedura StretchBlt może jeszcze skalować wklejany obraz :

```
StretchBlt Picture2.hdc, 0, 0, 200, 200, Picture1.hdc, 0, 0, 100, 100, &HCC0020
```

Teraz do Picture2 wkleimy obszar 200x200 z Picture1 100x100 czyli otrzymamy rozciągnięcie obrazu. Możemy też tworzyć maskę dla obrazków zmieniając końcowy parametr &HCC0020 na następujące :

```
BitBlt(Picture3.hDC, 0, 0, 100, 100, Picture1.HDC, 0, 0, &HBB0226)
```

```
BitBlt(Picture3.hDC, 0, 0, 100, 100, Picture2.HDC, 0, 0, &H8800C6)
```

Picture1:



Picture2:



W ten sposób na Picture3 zostaną wklejone wszystkie piksele z Picture2 które mają czarny odpowiednik w Picture1. Te zaś które jako odpowiednik w Picture1 mają kolor biały nie zostaną wklejone. Dzięki temu pozbywamy się tła w obrazkach.

## 55. Operacje na polu tekstowym

<http://visual.basic.kaem.forall.pl/>

### 1. Dopisywanie tekstu w wybranym miejscu

Jeżeli chcemy w danym miejscu w tekście dopisać jakiś ciąg znaków nie musimy używać wycinania tekstu, wstawiania i ponownego wklejania, można użyć dwóch funkcji:

`Text1.SelStart` ' ustawia pozycję kursora w wybranym miejscu

`Text1.SelText` ' wstawia tekst w wybranej pozycji

### 2. Sprawdzenie czy pole tekstowe uległo zmianie

Wystarczy użyć prostej obsługi zdarzenia:

```
Private Sub Text1_Change()
```

```
End Sub
```

### 3. Zazaczenie tekstu

Aby zaznaczyć tekst w wybranym miejscu wystarczy ustawić kursor w wybranej pozycji a następnie wybrać ile znaków ma być zaznaczonych.

`Text1.SelStart = 2` ' ustawia kursor na pozycji 2 znaku

`Text1.SelLength = 5` ' zaznacza 5 znaków

Aby zaznaczyć wszystko wystarczy tylko pobrać długość tekstu i ustawić kursor w pozycji 0:

```
Text1.SelStart = 0
```

```
Text1.SelLength = Len(Text1.Text)
```

#### 4. Znak nowego wiersza (gdy włączone Multiline)

Aby uzyskać znak nowego wiersza należy do swojego tekstu dodać dwa znaki z tablicy ascii, są to Chr(13) oraz Chr(10):

```
Text1.Text = Text1.Text & chr(13) & chr(10) & "Nowy tekst"
```

## 56. Iif

<http://visual.basic.kaem.forall.pl/>

Iif jest to funkcja sprawdzająca warunek i zwracająca różne wartości w przypadku gdy warunek został spełniony lub gdy nie został. Wygląda to tak :

```
Rezultat = Iif(warunek,gdy true, gdy false)
```

Przykładowe użycie

```
Wyplata = 1000  
Czyok = Iif(Wyplata>900,"OK","za malo")
```

Zmienna Czyok będzie wynosiła OK ponieważ Wyplata jest większa niż 900, w innym wypadku wynosiła by "za malo". Funkcja jest bardzo przydatna i pozwala na uproszczenie kodu, nie stosując jej musielibyśmy napisać :

```
Wyplata = 1000  
If Wyplata > 900 Then  
Czyok = "OK"  
Else  
Czyok = "za malo"  
End If
```

Lub :

```
Wyplata = 1000  
If Wyplata > 900 Then Czyok = "OK"  
If Wyplata < 901 Then Czyok = "za malo"
```

## 57. Split i Join

<http://visual.basic.kaem.forall.pl/>

### Split

Split jest funkcją która może podzielić wybrany tekst na części, miejscem oddzielenia jest wybrany przez nas znak/ciąg znaków. Załóżmy że chcemy z jakiegoś tekstu wydobyć osobno wyrazy:

```
Tekst = "to jest jakiś tekst"  
Wyraz = Split(Tekst," ")
```

Prawidłowo należy wcześniej zadeklarować tablicę dynamiczną:

```
Dim Wyras() As String
Tekst = "to jest jakiś tekst"
Wyras = Split(Tekst," ")
```

Powstanie nam zmienna Wyras w postaci tablicy, gdzie każdy oddzielony wyraz ma odpowiedni index, postaną następujące wartości

```
Wyras(0) = "to"
Wyras(1) = "jest"
Wyras(2) = "jakiś"
Wyras(3) = "tekst"
```

Żeby dowiedzieć się ile elementów w tablicy powstało wystarczy użyć polecenia sprawdzającego ilość elementów z tablicy

```
Ilosc = Ubound(Wyras)
```

### Join

Join jest funkcją odwrotną do split, łączy on zmienne z wybranej tablicy oddzielający wybranym przez nas znakiem/ciągiem znaków.

```
Wyras(0) = "AA"
Wyras(1) = "BB"
Wyras(2) = "CC"
Tekst = Join(Wyras,"+")
```

Tekst będzie zawierał wartość: "AA+BB+CC". Przydaje się to gdy np. chcemy stworzyć baze danych opartych na .txt i chcemy zapisać wiele wartości w jednej linijce.

## 58. Type

<http://visual.basic.kaem.forall.pl/>

Instrukcja Type pozwala nam na zadeklarowanie własnych zestawów zmiennych. Jest to bardzo przydatne przy większych projektach, pozwala na łatwiejsze orientowanie się w kodzie oraz upraszcza pewne rzeczy. Chodzi o to że możemy stworzyć kilka zmiennych i przyporządkować je do obiektu Type. Wszystko wygląda mniej więcej tak:

```
Private Type Nazwa
- zmienne -
End Type
```

Oczywiście jeżeli chcemy stworzyć Type publicznie zamieniamy Private na Public. Zadeklarowanie Typu z trzema zmiennymi można przedstawić w następujący sposób:

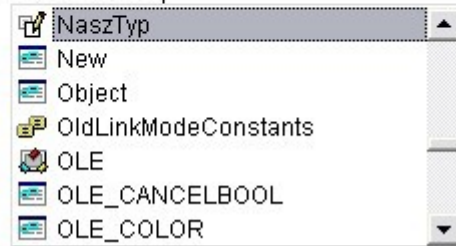
```
Private Type NaszTyp
PozycjaX As Integer
PozycjaY As Integer
Wartosc As String
End Type
```



Teraz aby wszystko działało należy zadeklarować w dowolny sposób zmienną jako NaszTyp

```
Private Type NaszTyp
    PozycjaX As Integer
    PozycjaY As Integer
    Wartosc As String
End Type
```

```
Private Zmienna As NaszTyp|
```



Odrzu można zauważyć że 'NaszTyp' pojawił się w podręcznej liście Typów dla zmiennych itp.

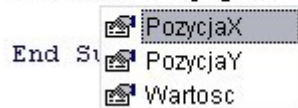
Po zadeklarowaniu takiej zmiennej możemy korzystać z trzech wartości:

[Zmienna.PozycjaX](#)  
[Zmienna.PozycjaY](#)  
[Zmienna.Wartosc](#)

Oczywiście korzystanie z nich również jest ułatwione ( Rozwijana lista z nazwami ) przez co nie musimy głowiąć się jak dokładnie nazywa się nasza zmienna:

```
Private Sub Form_Load()
```

```
Zmienna.PozycjaX
```



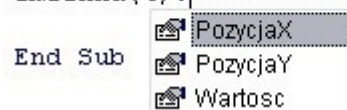
```
End Sub
```

Można także deklarować zmienne naszego typu w postaci tablicowej, nie chodzi mi tutaj tylko o deklaracje w Type ale także o deklaracje samych zmiennych naszego typu:

```
Private Zmienna(10) As NaszTyp
```

```
Private Sub Form_Load()
```

```
Zmienna(4) .|
```



```
End Sub
```

Co to wszystko nam daje ? Możliwość tworzenia zmiennych które dzięki różnym przechowywanym wartościom bardzo mogą przypominać korzystanie z obiektów. Dzięki temu można usprawnić sortowanie (nie trzeba by korzystać z tablic dwu wymiarowych i zamieniać każdego elementu), wyszukiwanie danych, wypisywanie danych i wiele innych rzeczy.

Przykład zastosowania (część bazy adresów):

```
Private Type Informacje
```

```
Ulica As String
```

```
Miasto As String
```

```
Kod As String
```

```
Telefon As String
```

```
End Type
```

```
Private Dane(2) As Informacje
```

```
Private Sub Form_Load()
```

```
Dane(0).Kod = "41-200"
```

```
Dane(0).Miasto = "Sosnowiec"
```

```
Dane(0).Telefon = "brak"
```

```
Dane(0).Ulica = "Jasna 12a"
```

```
Dane(1).Kod = "41-700"
```

```
Dane(1).Miasto = "Jaworzno"
```

```
Dane(1).Telefon = "brak"
```

```
Dane(1).Ulica = "brak informacji"
```

```
Dane(2).Kod = "42-220"
```

```
Dane(2).Miasto = "Zabrze"
```

```
Dane(2).Telefon = "brak"
```

```
Dane(2).Ulica = "Krótka 15/106"
```

```
End Sub
```

Gdybyśmy chcieli posegregować taką bazę (obojętnie jakim sposobem) używając zamiast Type tablic, musielibyśmy przy każdej zmianie elementów tablicy zmieniać pokolei wszystkie dane. Dzięki Type gdy chcemy zmienić wszystkie dane, i np przesunąć Wartości z Dane(2) do Dane(0) wystarczy użyć jednego przypisania:

```
Dane(0) = Dane(2)
```

Dodatkowo każda ze zmiennych w naszym type może mieć postać tablicy:

```
Private Type Informacje
```

```
Ulica As String
```

```
Miasto As String
```

```
Kod As String
```

```
Telefon As String
```

```
Notatka(10) As String
```

```
End Type
```

```
Private Dane(2) As Informacje
```

```
Private Sub Form_Load()
```

```
Dane(0).Notatka(0) = "Notatka 1"
```

```
Dane(0).Notatka(1) = "Notatka 1"
```

```
End Sub
```

W ten sposób można zapisać aż 30 notatek (do każdego elementu Dane(numer) po 10 różnych danych).

Zmienne zadeklarowane w Type mogą przyjąć także (oprócz normalnych typów i tablic) postać zmiennych stworzonych przez inne Type.

Private Type Informacje

Ulica As String

Miasto As String

Telefon As String

End Type

Private Type GlowneDane

Imie As String

Nazwisko As String

Adres As Informacje

End Type

Private Osoba(2) As GlowneDane

Private Sub Form\_Load()

Osoba(0).Adres.Miasto = "Sosnowiec"

Osoba(0).Adres.Telefon = "260-16-25"

Osoba(0).Adres.Ulica = "Jasna 12a"

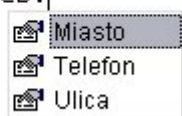
Osoba(0).Imie = "Jan"

Osoba(0).Nazwisko = "Kowalski"

End Sub

Widać tutaj że każda Osoba ma 3 rodzaje danych ( Adres, Imie, Nazwisko ), natomiast Każdy Adres ma także 3 rodzaje danych ( Miasto, Telefon, Ulica ). Takie coś powstało dzięki zadeklarowaniu Adres As Informacje. Wszystko jest dalej wyświetlane w przejrzystej liście:

```
Private Sub Form_Load()  
Osoba(0).Adres.  
End Sub
```



Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 59. With

<http://visual.basic.kaem.forall.pl/>

With jest elementem języka Visual Basic który pozwala na prostą i szybką zmianę kilku właściwości dla obiektu lub zmiennej typu użytkownika (type).

Używanie jest bardzo proste, wybieramy obiekt lub naszą zmienną dla której chcemy zmienić właściwości a następnie wypisujemy je po kropce. Przykład:

```

With Command1
.Caption = "Tekst"
.Left = 100
.Top = 100
End With

```

Ogólny zarys wygląda tak:

```

With JakiśObiekt
// zmiana właściwości
End With

```

Właściwości poprzez With możemy także zmienić dla zmiennych stworzonego przez nas (tylko w wersji .php/.html) [Typu](#).

```

Private Type Test
A As Byte
B As Byte
C As Byte
End Type

```

```
Private Zmienna As Test
```

```
Private Sub Form_Load()
```

```

With Zmienna
.A = 5
.B = 8
.C = 50
End With

```

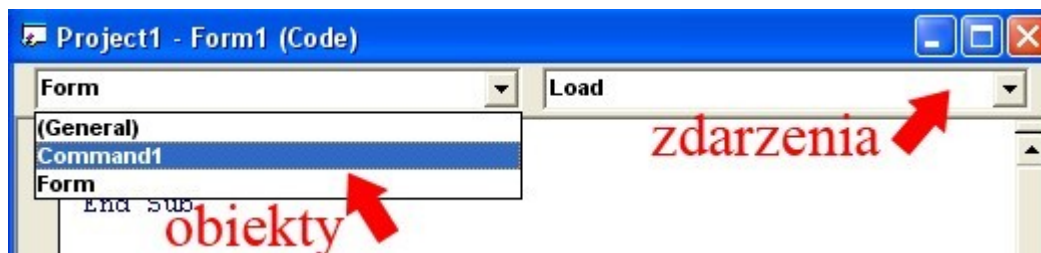
```
End Sub
```

Zapis With uprasza kod, dzięki temu przejrzystość programu jest większa, także program powinien działać minimalnie szybciej (w teorii, w praktyce sprawdzając czas zmiany 10 właściwości 1 obiektu 10000 razy poprzez with i bez używania with czas potrzebny na operacje with był tylko o 1% krótszy).

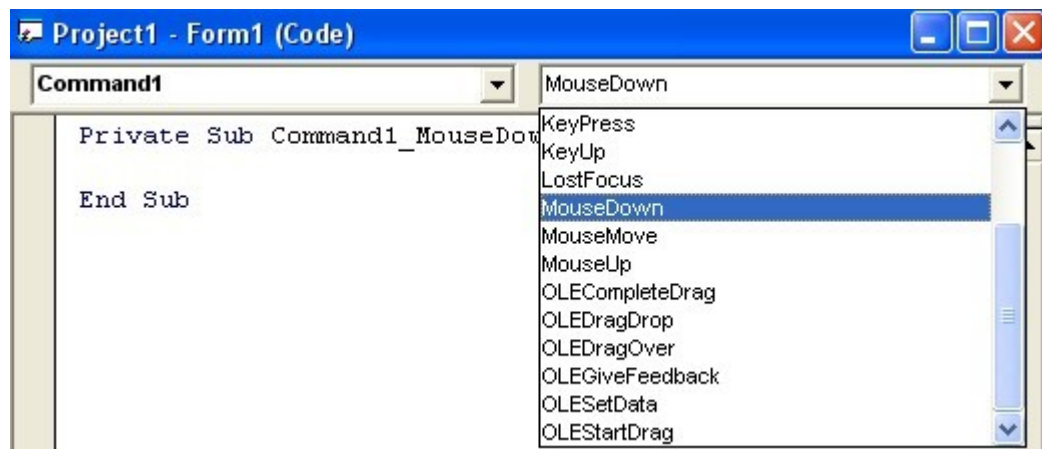
## 60. Zdarzenia

<http://visual.basic.kaem.forall.pl/>

Visual basic oferuje bardzo duży zestaw zdarzeń dla wybranych obiektów. Aby zobaczyć jakie zdarzenia są dostępne zagłębiamy w kod źródłowy formularza na którym aktualnie pracujemy.



Każdy obiekt ma swoje zdarzenia. Większość z nich się powtarza. Aby stworzyć procedurę obsługującą wybrane zdarzenie należy wybrać jedno ze zdarzeń a kond na formie pojawi się automatycznie.



#### Podstawowe zdarzenia:

**Click()** - kliknięcie w wybrany obiekt

**DbClick()** - podwójne kliknięcie

**DragDrop(Source As Control, X As Single, Y As Single)** - gdy na obiekt 'upuszczimy' inny obiekt. Zmienna Source przechowuje nazwę obiektu, X i Y miejsce w którym został upuszczony

**DragOver(Source As Control, X As Single, Y As Single, State As Integer)** - jak wyżej tylko że wystarczy że najedziemy przeciąganym obiektem, nie musimy go upuszczać. Dodatkowo dostępny jest status przeciąganego obiektu

**GotFocus()** - gdy obiekt dostanie fokus (stanie się aktywny i używany w obecnej chwili)

**KeyDown(KeyCode As Integer, Shift As Integer)** - gdy wciśniemy jakiś klawisz, KeyCode to kod klawiszu, Shift ma wartość 0 gdy klawisz shift nie jest wciśnięty i 1 gdy jest

**KeyPress(KeyAscii As Integer)** - tak samo jak w poprzednim przykładzie tyle że nie dostajemy informacji o klawiszu Shift, a zwrócony kod klawisza dostajemy w postaci numeru z tablicy Ascii

**KeyUp(KeyCode As Integer, Shift As Integer)** - podobne zdarzenie jak KeyDown tyle że zachodzi w momencie gdy wciskamy przycisk (KeyDown gdy puszczaemy)

**LostFocus()** - gdy obiekt traci fokus (przestaje być aktywny i używany, lub gdy fokus dostanie inny element)

**MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)** - gdy klikniemy myszką w wybrany obiekt (Button przechowuje numer przycisku którym klikamy, X i Y to pozycje kursora względem obiektu)

**MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)** - jak poprzednio tyle że zdarzenie zachodzi w momencie gdy przejeżdżamy myszką na obiektem (nie musimy klikać, wystarczy na obiekt najechać)

**MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)** - zdarzenie podobne do MouseDown tyle że zachodzi w momencie upuszczenia klawiszu myszki

**Validate(Cancel As Boolean)** - podobne zdarzenie do LostFocus

Niektóre zdarzenia wybranych obiektów:

- **Form:**

**Load()** - gdy forma jest ładowana

**Resize()** - gdy zmieniany jest rozmiar formy

**Unload(Cancel As Integer)** - gdy forma jest wyłączana

- **TextBox**

**Change()** - gdy wartość pola została zmieniona

- **ComboBox**

**Scroll()** - gdy lista jest przewijana

**Przykład:**

Na formie umieszczamy jeden TextBox (o nazwie Text1) i tworzymy dla niego zdarzenia

```
Private Sub Text1_Change()  
MsgBox "zmieniłeś coś"  
End Sub
```

```
Private Sub Text1_Click()  
MsgBox "klikłeś w text1"  
End Sub
```

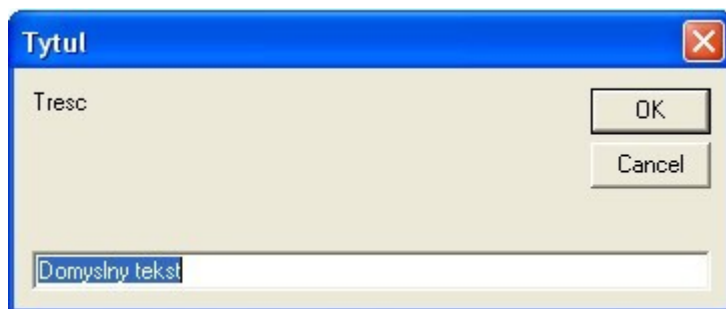
## 61. Input Box

<http://visual.basic.kaem.forall.pl/>

**InputBox** jest okienkiem pobierającym tekst. Jest bardzo przydatne gdy chcemy pobrać od użytkownika różne informacje i nie musimy dzięki temu pisać osobnych formów do pobierania danych.

Stosowanie:

```
Co = InputBox("Tresc", "Tytul", "Domyslny tekst")
```



Po użyciu pod zmienną Co zyskujemy odpowiedź użytkownika.

W przypadku gdy użytkownik wciśnie 'Cancel' odpowiedź będzie pusta.

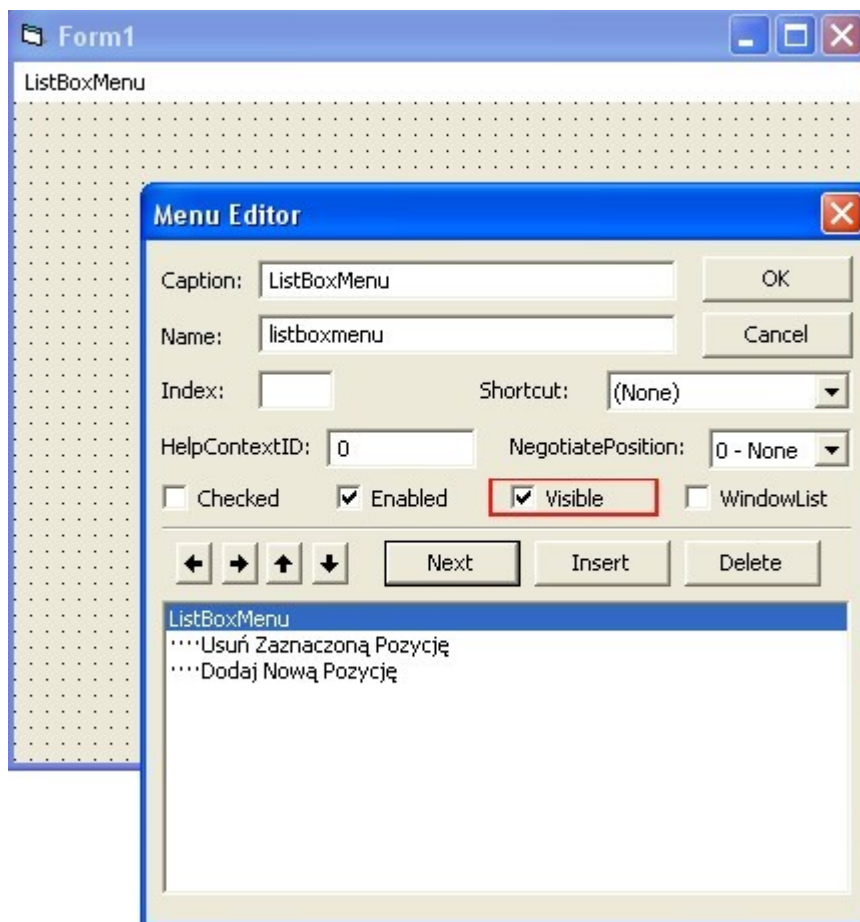
## 62. Menu Podręczne

<http://visual.basic.kaem.forall.pl/>

Menu podręczne jest dużym ułatwieniem dla użytkownika projektowanej przez nas aplikacji.

Pierwszą czynnością jaką musimy zrobić jest zaprojektowanie menu w menu editorze.

Stwórzmy proste menu które pozwoli użytkownikowi dodawać i usuwać pola z ListBox-a.



Nasze menu musi być pozornie ukryte, dlatego odchaczamy Visible z pozycji ListBoxMenu

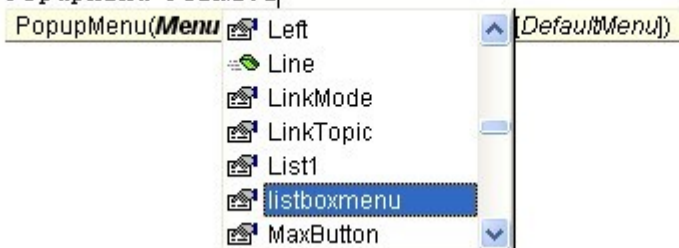
tak żeby kwadracik przy Visible był pusty.

Po tej czynności menu zniknie z formy, możemy je natomiast wywołać w dowolnym momencie:

**PopupMenu NazwaFormy.NazwaMenu**

Naszemu menu nadaliśmy wartość Name: listboxmenu. Wywołajmy je w momencie gdy użytkownik kliknie prawym myszy gdzieś na obiekcie ListBox1

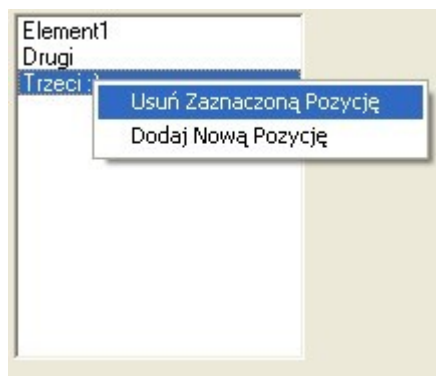
```
Private Sub List1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 2 Then PopupMenu Form1.listboxmenu
End Sub
```



```
Private Sub List1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 2 Then PopupMenu Form1.listboxmenu ' 1
End Sub
```

1. Jeżeli wciśniętym przyciskiem myszki jest przycisk nr.2 to wyświetl menu

Teraz gdy użytkownik wciśnie 2 przycisk myszki w obiekcie List1 otrzyma



Wystarczy już tylko zaprogramować przyciski z menu. Jeżeli nasz przycisk z menu ma Name: przycisk1, to piszemy dla niego zdarzenie:

```
Private Sub przycisk1_Click()
```

```
End Sub
```

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)



## 63. Option Explicit

<http://visual.basic.kaem.forall.pl/>

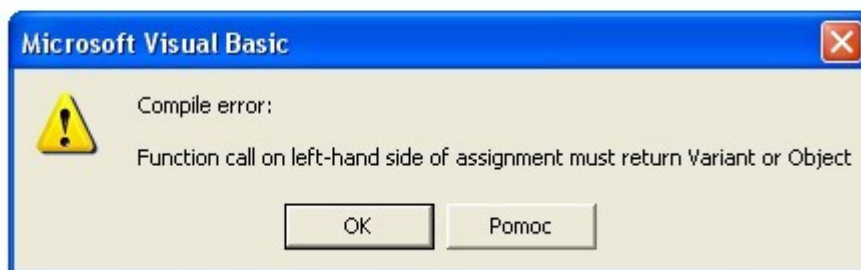
Umieszczenie na samym początku kodu lub zaraz po deklaracji zmiennych Option Explicit spowoduje że kompilator będzie informował o źle lub wogóle nie zadeklarowaniu zmiennych.

Deklarowanie zmiennych jest na tyle ważne że w niektórych wersjach systemu Windows program w którym używamy zmiennych bez ich deklaracji może nie działać prawidłowo (na około 5% systemów windows pojawia się taki problem, zależy to od konkretnej wersji oraz zainstalowanych patchów)

Informacje o zmiennych których nie zadeklarowaliśmy pojawiają się dopiero podczas używania programu (nie podczas ani przed kompilacją (np. dla przycisku w którym byśmy wywołali nieistniejącą zmienną błąd się pojawi dopiero gdy klikniemy w ten przycisk), chociaż z drugiej strony komunikat przybiera nazwę 'Compile error:' ?), dlatego testując program pasowało by przetestować wszystkie jego opcje.

### Option Explicit

```
Private Sub Form_Load()  
Cos = 5  
MsgBox Cos  
End Sub
```



Na rysunku powyżej pojawił się błąd. Aby go naprawić wystarczy zadeklarować zmienną.

### Option Explicit

```
Private Sub Form_Load()  
Dim Cos As Byte  
Cos = 5  
MsgBox Cos  
End Sub
```

## 64. Praktyka – Zmienne

<http://visual.basic.kaem.forall.pl/>

### Zmienne liczbowe

Na zmiennych liczbowych można wykonywać działania matematyczne, używa się do tego następujących symboli:

- +      dodawanie
- odejmowanie
- \*      mnożenie

/        dzielenie  
\  
Mod    reszta z dzielenia  
^       potęgowanie

W przykładzie wygląda to następująco :

```
Liczba1 = 16  
Liczba2 = 14  
Wynik = Liczba1 + Liczba2
```

Oczywiście można wykonywać wiele operacji w jednej linii i porządkować dane operacje nawiasami:

```
Liczba1 = 4  
Liczba2 = 6  
Liczba3 = 3  
Wynik = (Liczba1 + Liczba2) * Liczba3
```

W działaniach nie ma pierwiastkowania, to dlatego że tak naprawdę pierwiastkowanie jest podnoszeniem liczby do potęgi mniejszej niż 1. Aby otrzymać pierwiastek drugiego stopnia podnosimy liczbę do potęgi 1/2, trzeciego stopnia 1/3, czwartego 1/4... itd.

```
Dim Liczba As Byte  
Liczba = 9  
MsgBox 9 ^ (1 / 2)
```

W komunikacie otrzymamy liczbę 3.

### Typy zmiennych liczbowych w praktyce

Deklarując zmienne należy pamiętać o ich zasięgach, chcąc więc wykonać jakieś działania na liczbach należy się najpierw zastanowić jaki typ liczb będzie potrzebny. Jeżeli wykroczymy poza zakres typu zmiennej otrzymamy błąd.

```
Dim A As Byte  
A = 4000 * 5
```

W takim wypadku należało by użyć zmiennych typu integer, ale co zrobić jeżeli nie wiemy dokładnie na jakich liczbach będziemy operować, niektórzy omijają deklaracje zmiennych. Jest to równoznaczne z zadeklarowaniem zmiennej jako Variant. Obciążeniem jest oczywiście to że taka liczba zajmuje trochę bajtów w pamięci.

Kolejnym problemem spotykanym tylko na niektórych komputerach z winxp (5%) jest wykonanie operacji na innych zmiennych niż ta przypisana. Ma to taką postać:

```
Dim A As Byte  
Dim B As Integer  
A = 40  
B = A * 110
```

Zapis wydaje się poprawny i będzie działał poprawnie na większości komputerów, jednak prawidłowy zapis powinien wyglądać następująco:

```
Dim A As Byte
Dim B As Integer
Dim AOK As Integer
A = 40
AOK = Int(A)
B = AOK * 110
```

### Zmienne tekstowe

Zmienne tekstowe można dowolnie łączyć oraz obrabiać przez różne funkcje, np ( tylko w wersji .php/.html ) takie, łączenie zmiennych tekstowych wygląda następująco:

```
Tekst1 = "AB"
Tekst2 = "CD"
TekstRazem = Tekst1 & Tekst2
```

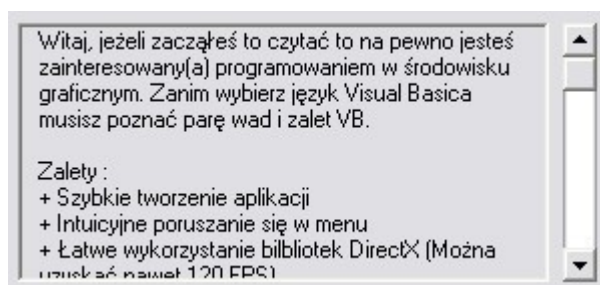
Aby dodać nowy tekst do istniejącego wystarczy napisać to tak:

```
Tekst = "AB"
Tekst = Tekst & "CD"
```

## 65. Praktyka – Suwaki

<http://visual.basic.kaem.forall.pl/>

Okazuje się że suwaki w praktyce używane są głównie do określania wartości, warto jednak przyjrzeć się temu co należy zrobić aby wykorzystać je do przeciągania pól tekstowych. Umieścimy na formie PictureBox, włożymy do niego Label i postawmy obok (poza PictureBoxem) suwak pionowy. Teraz trzeba wkleić do Labela trochę tekstu, najlepiej tak żeby nie mieściło się to w PictureBoxie, powinno to wyglądać tak:



Teraz należy ustawić ScaleMode w PictureBoxie i na Formie na Pixel. Najważniejsze czego potrzebujemy to dowiedzieć się ile tekstu nie mieści się w PictureBoxie, w tym celu należy od wysokości tekstu odjąć wysokość PictureBox i dodatkowo odjąć 4 piksele (obwódka PictureBoxa). Następnie należy ustalić wskaźnik przesunięcia, ja polecam 10. Teraz wysokość tekstu której nie możemy wyświetlić dzielimy przez 10 i przypisujemy jako maksymalną wartość suwaka. Jeszcze tylko wystarczy dopisać zdarzenie do suwaka i wszystko będzie się ładnie przesunęło, wyjaśni to poniższy przykład:

```
Private Sub Form_Load()
WysokoscTekstu = Label1.Height
IleSieWyswietli = Picture1.Height - 4
IleSieNieWyswietli = WysokoscTekstu - IleSieWyswietli ' obliczamy ile tekstu się nie
zmieści
VScroll1.Max = Round(IleSieNieWyswietli / 10) ' ustalamy maksymalną wartość suwaka
```

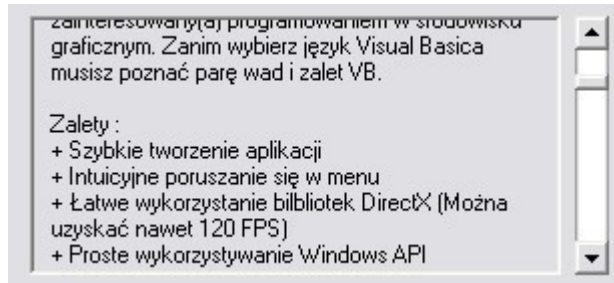
End Sub

Private Sub VScroll1\_Change()

Label1.Top = -VScroll1.Value \* 10 ' przesuwamy tekst względem wartości suwaka

End Sub

Nasz suwak działa, powinno to wyglądać następująco:



## 66. Praktyka - Używanie kolorów

<http://visual.basic.kaem.forall.pl/>

### RGB

Kolor w postaci RGB jest składową trzech barw (R - red/czerwony, G - green/zielony, B - blue/niebieski).

Podstawowe kolory z palety kolorów w Visual Basicu występują w postaci &H00BBGGRR&

Wartość RR, GG i BB to wartość w trybie szesnastkowym zapisana w postaci dwóch znaków. Maksymalnie może więc osiągnąć wartość 255, minimalnie oczywiście zero.

Mieszanie kolorów jest rzeczą dość prostą, jest identyczne do mieszania barw światła. Jeżeli zmieszamy światło czerwone, zielone i niebieskie otrzymamy światło białe, jeżeli nie użyjemy żadnego światła nie uzyskamy nic, czyli kolor czerni.

Tak więc żeby uzyskać kolor biały należy użyć wszystkich kolorów w maksymalnym nasyceniu (255) czyli w postaci szesnastkowej FF. Kolor taki będzie wyglądał następująco: &H0FFFFFFF&

W Visual basicu istnieje bardzo pomocnicza funkcja która pozwala zamienić kolor zbudowany z różnych wartości 3 barw od razu na kod przyjmowany przez vb. Jest to funkcja RGB(czerwony,zielony,niebieski). Zauważmy że w ułożeniu &H00BBGGRR& kolory te występują w odwrotnej kolejności. Do funkcji RGB nie należy używać zamieniania kolorów z wartości 0-255 na postać szesnastkową. Wygląda to tak:

Czerwony = RGB(255,0,0)

CiemnyCzerwony = RGB(100,0,0)

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 67. Graficzny pasek postępu

<http://visual.basic.kaem.forall.pl/>

Pasek postępu jest bardzo przydatny. Szczególnie w wypadku gdy użytkownik musi długo czekać na wykonanie jakichś operacji. Można użyć paska postępu wbudowanego w comctl32.ocx (Microsoft Windows Common Controls 5.0 (SP2)), jednak w takim wypadku nasz pasek stałby się schematyczny, a do naszego programu pasowałoby dołączyć plik comctl32.ocx który zajmuje trochę KB.

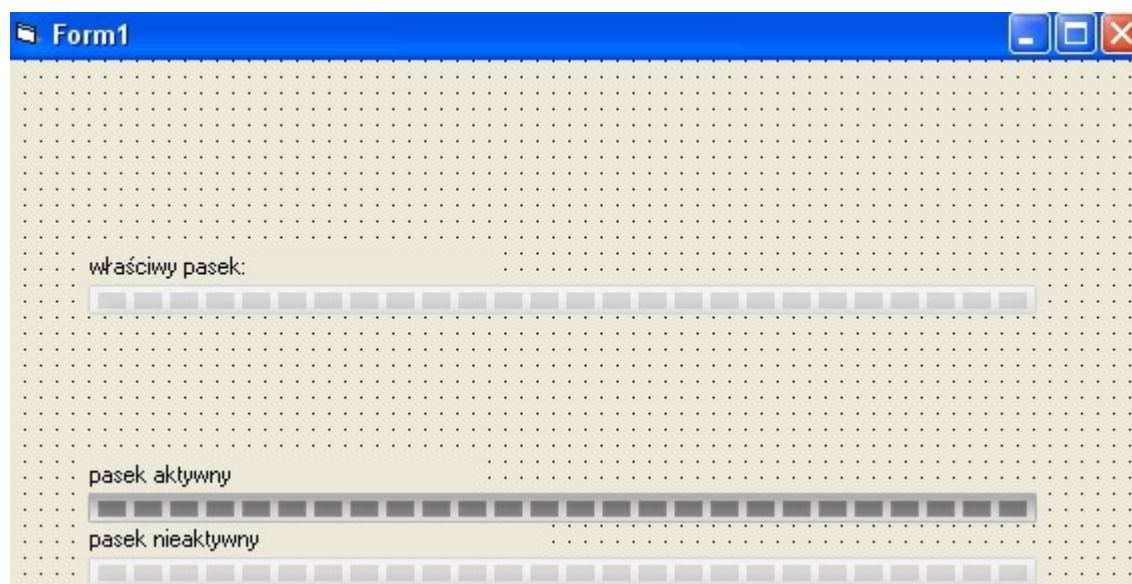
Bardziej wydajniejsze jest stworzenie własnego paska postępu opartego na 3 PictureBoxach. Pierwsze co musimy przygotować to dwa pliki graficzne o wybranej przez nas wielkości. Jeden plik graficzny będzie paskiem nieaktywnym, drugi paskiem aktywnym.



W opisywanym przypadku pasek aktywny został umieszczony jako Picture1, a pasek nieaktywny jako Picture2.

Ważną cechą jest długość paska, założmy że użytkownik będzie nadawał dla naszego paska wartości od 0 do 100. Jeżeli nasz pasek ma 450 pikseli, wartość paska aktywnego będzie równa wartości przypisanej przez użytkownika razy 4.5.

Na formie umieścimy trzecie PictureBox (nazwijmy go po prostu Pasek) i wklejmy do niego pasek nieaktywny.



Pierwszą czynnością jaką musimy wykonać jest napisanie procedury kopiującą wybrane części naszego paska. Do kopiowania grafiki użyjemy funkcji (tylko w wersji .php/.html) BitBlt. Nasza procedura powinna kopiować wybrane części paska aktywnego do naszego paska właściwego. Od razu narzuca się myśl aby zaoszczędzić trochę na pamięci, nie ma sensu kopiować już wcześniej skopiowanych obszarów paska. Tak więc nasze kopiowanie powinno zaczynać się od wybranego miejsca i kończyć w pewnym wybranym miejscu.

```
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal X As Long,
ByVal Y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long,
ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
```

```
Private Sub Kopiuj(ByVal OdC As Integer, ByVal DoC As Integer)
BitBlt Pasek.hDC, OdC, 0, DoC - OdC, 15, Picture1.hDC, OdC, 0, &HCC0020
End Sub
```

Stworzona procedura będzie kopiowała elementy Picture1 (paska aktywnego) do paska głównego, zaczynając od wartości OdC a kończąc na DoC. Gdy wartość paska wynosi np. 30 a my chcemy ją zmienić na 50 będziemy kopiowali tylko wybrany obszar, dlatego potrzebna nam zmienna przechowująca wartość paska.

**Private Skopiowano As Integer**

Została do napisania jeszcze jedna procedura, przeliczająca wartość od 0 do 100 na wartość długości paska (czyli mnożąc przez wybrany współczynnik). Automatycznie można dodać do tej procedury odniesienie do procedury kopiowania paska.

```
Private Sub Wartosc(ByVal Ile As Byte)
Dim Ile2 As Integer
Ile2 = Round(Ile * 4.5)
Kopiuj Skopiowano, Ile2
Skopiowano = Ile2
End Sub
```

Mnożymy wartość przez 4.5 ponieważ szerokość stworzonej przez nas grafiki to 450 pikseli.

Nasz pasek jest już praktycznie gotowy, teraz aby ustalić dla niego wybraną wartość wystarczy użyć wywołania procedury Wartosc:

**Wartosc 60**

Użycie Wartosc 60 'ustawi' nasz pasek na 60% postępu. Zostaje już tylko dodanie procedury czyszczącej pasek (czyli stworzenie zwykłego przekopiowania obrazka Picture2 do naszego pola Pasek). Można to napisać na procedurze BitBlt lub po prostu przypisując wartości Picture.

```
Private Sub Zeruj()
Pasek.Picture = Picture2.Picture
Skopiowano = 0
End Sub
```

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

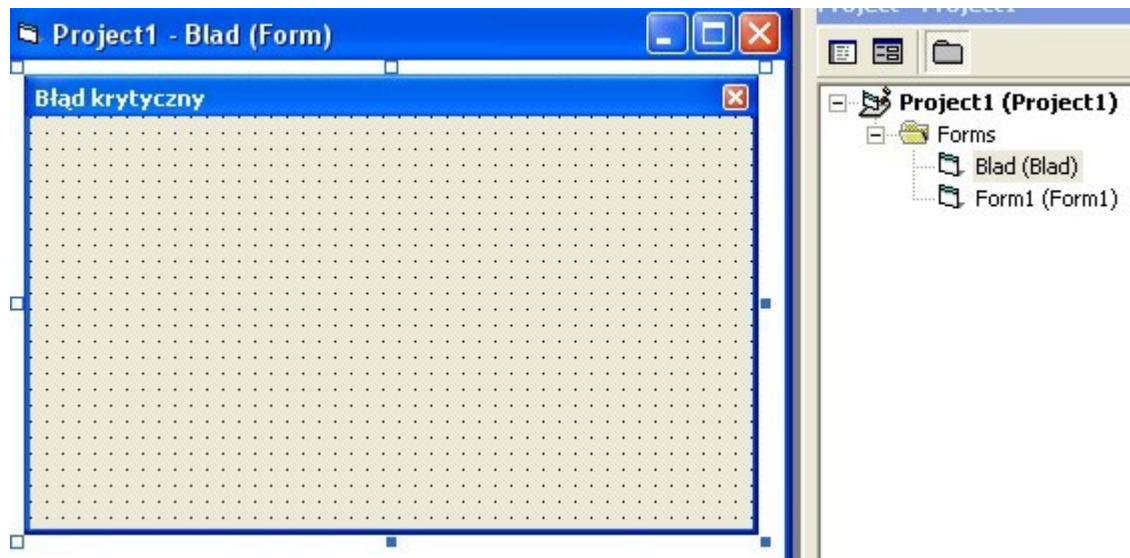
## 68. Obsługa błędów + http

<http://visual.basic.kaem.forall.pl/>

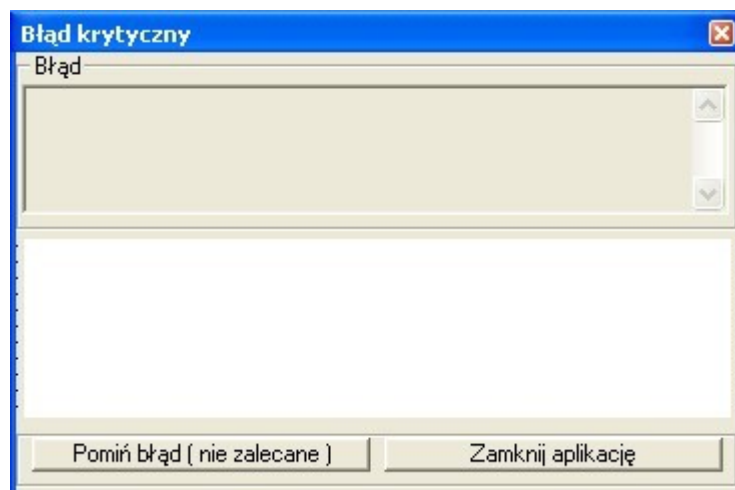
Opisany tutaj program i skrypty pozwolą na dostarczenie informacji o błędach na stronie autora. W ten sposób programista będzie mógł zobaczyć jakie najczęstsze błędy są

zgłaszane przez użytkowników, przeczytać ich komentarze i naprawić usterki w swoim programie :)

Stwórzmy projekt, dodajmy do niego drugą Formę. Będzie ona naszym okienkiem błędu. Zmieńmy nazwę tego okienka na ' Błąd krytyczny', ustawmy Name = Bład, BorderStyle = 4 i ScaleMode na 3.



Teraz w okienku trzeba ułożyć kilka rzeczy. Najważniejsze to pole tekstowe 'Text1' z ustawionym Multiline na True i pionowym suwakiem, dwa przyciski 'Command1' oraz 'Command2', kontrolkę WebBrowser 'WebBrowser1' ( Project > Components > Microsoft Internet Controls ) i jeden obiekt Frame 'Frame1'.



Wypadało by też aby nasze okienko pojawiała się na środku ekranu ( StartUpPosition = 2 ).

Nasze okienko błędu będzie się pojawiało automatycznie gdy w jakiejś procedurze w Form1 pojawi się błąd, trzeba więc stworzyć funkcję wywołującą nasze okno, zrobmy to dla pola przycisku. Umieścmy pole przycisku 'Command1' na Form1 i wpiszmy dla niego procedurę.

```
Private Sub Command1_Click()  
On Error GoTo BładW
```



```
Dim Wynik As Integer  
Wynik = 8 / 0
```

```
Exit Sub  
BladW:  
BladF
```

```
End Sub
```

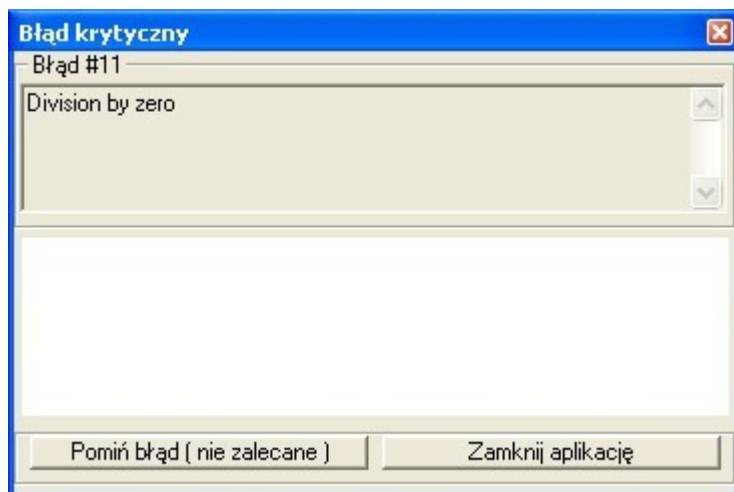
Program po pojawieniu się obojętnie jakiego błędu (w naszym przypadku dzielenia przez 0) będzie przenosił się do miejsca BladW a następnie wywoływał procedurę BładF, napiszmy ją:

```
Private Sub BladF()  
Form1.Enabled = False  
Blad.Show  
End Sub
```

Form1.Enabled = False zablokuje tymczasowo Form1. Przejdźmy teraz spowrotem do okienka Blad, należało by zrobić przy jego ładowaniu wyświetlanie bieżącego numeru błędu oraz komunikatu błędu:

```
Private Sub Form_Load()  
Frame1.Caption = "Błąd #" & Err.Number  
Text1.Text = Err.Description  
End Sub
```

Po uruchomieniu programu i wciśnięciu przycisku wywołującego dzielenie pojawi się nasze okienko:



Zaprogramujmy przyciski z formularza Blad. Stworzyliśmy dwa przyciski, jeden będzie wyłączał okienko błędu i uaktywniał spowrotem nasz Form1:

```
Private Sub Command1_Click()  
Form1.Enabled = True  
Unload Blad  
End Sub
```

Drugi przycisk będzie wyłączał cały program ( będzie trzeba w nim zrobić unload wszystkich form jakich używamy ):



```
Private Sub Command2_Click()
Unload Blad
Unload Form1
End Sub
```

Trzeba jeszcze dodać obsługę zdarzenia dla Unload formy Blad ( inaczej jak użytkownik kliknie [x] forma błędu zostanie wyłączona, a Form1 będzie nieaktywne:

```
Private Sub Form_Unload(Cancel As Integer)
Form1.Enabled = True
End Sub
```

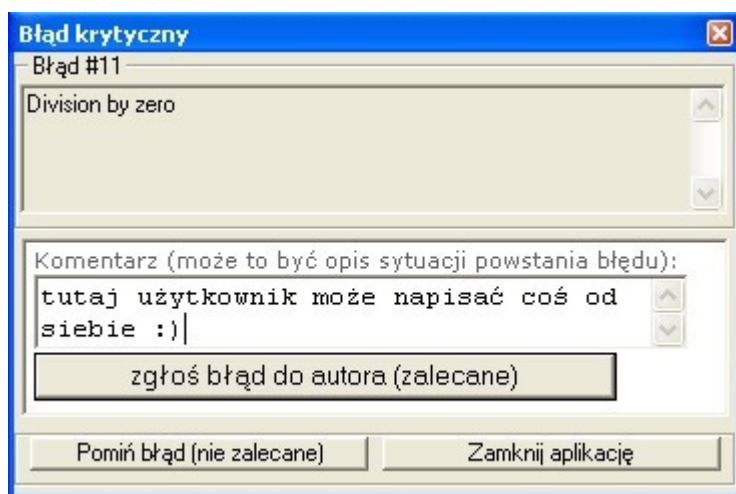
Po wprowadzeniu obsługi tego zdarzenia można usunąć Form1.Enabled = True ze zdarzenia Command1\_click.

Teraz zajmijmy się stworzeniem formularza w html-u który będzie pozwalał na to, aby użytkownik przesłał do nas treść błędu oraz mógł go opisać. Komunikat błędu będzie się wysyłał gdzieś na nasz serwer internetowy ( serwer musi obsługiwać PHP ).

Najpierw stwórzmy plik .html, oczywiście tworzymy go w procedurze ładowania formularza Blad:

```
Open "blad.html" For Output As #1
Print #1, "<style> font {color:#666666;text-decoration:none;font-family:Verdana;font-size:8pt;} body {margin-left:4;margin-top:2;margin-bottom:0} </style>";
Print #1, "<body scroll=no scrollbars=no><font>"
Print #1, "<form action='http://kaem.forall.pl/vb/bledy.php' method=post>";
Print #1, "<input type=hidden name=numerbledu value='' & Err.Number & ''>";
Print #1, "<input type=hidden name=blad value='' & Err.Description & ''>";
Print #1, "Komentarz (może to być opis sytuacji powstania błędu):<br><textarea cols=38 rows=2 name=opis></textarea><br>";
Print #1, "<input type=submit value='zgłoś błąd do autora (zalecane)'></form>";
Close #1
WebBrowser1.Navigate App.Path & If(Right(App.Path, 1) = "\", "", "\") & "blad.html"
```

Nie będę opisywał działania tego kodu, zainteresowanym polecam poczytać o formularzach w html-u. Opiszę tylko co ten kod robi. Otóż ten kod, stworzy plik .html z formularzem do wypełnienia. Następnie cała strona zostanie załadowana do WebBrowser1 i wyświetlona na formie Blad.



Ostatnią czynnością jaką musimy zrobić jest napisanie skryptu php który będzie

zapisywał dane otrzymane z formularza do jakiegoś pliku tekstowego lub bazy mysql.

Stwórzmy na dysku dwa pliki 'bledy.php' i 'bledy.txt'. W pierwszym pliku umieścimy prosty skrypt:

```
<style> font {color:#666666;text-decoration:none;font-family:Verdana;font-size:8pt;}
body {margin-left:4;margin-top:2;margin-bottom:0} </style><body scroll=no
scrollbars=no><font>
<?
$file=fopen("bledy.txt","a");
fwrite($file,"$numerbledu|$blad|$opis\n");
fclose($file);
?>
Zgłoszono błąd, można zamknąć okno błędu
```

Drugi plik zostawmy pusty, będą do niego dopisywane błędy. Wszystko już prawie skończone, teraz trzeba jeszcze tylko wgrać pliki 'bledy.php' i 'bledy.txt' na serwer ( pogrubiony adres w zapisie formularza ) i zmienić chmod pliku 'bledy.txt' na 666.

Od tej pory wchodząc pod adres <http://kaem.forall.pl/vb/bledy.txt> będzie można zobaczyć spis zgłoszonych błędów.

Wystarczy tylko mieć serwer php i zmienić adres http na swój własny.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 69. WinSock

<http://visual.basic.kaem.forall.pl/>

Opisze tutaj najważniejsze informacje do połączenia z innym komputerem przy użyciu kontrolki Winsock, jeżeli takowej nie mamy wystarczy sobie wyszukać na google plik mswinsck.ocx.

Pierwsze co należy zrobić to umieścić na formie naszą kontrolkę, wybieramy z menu Project>Components i na liście powinniśmy znaleźć Microsoft Winsock Control 6.0, lub to samo z innym numerem wersji kontrolki.

Najważniejszymi funkcjami kontrolki są:

**RemoteHost** - określa host z którym mamy się połączyć

**RemoteHostIP** - jak wyżej ale tylko gdy znamy ip, poprzednia funkcja pozwala łączyć się z nazwami hostów

**RemotePort** - port na którym się łączymy

**State** - stan połączenia, najważniejsze to : 0 - nie jesteśmy połączeni, 7 - jesteśmy

**Listen** - ustawia naszą kontrolkę w stan nasłuchiwanie na połączenie

**Accept** - akceptuje połączenie

**Close** - zamyka połączenie

**Connect** - otwiera połączenie z hostem podanym w RemoteHost i portem z RemotePort lub z hostem i portem podanym przy Connect

**SendData** - wysyła dane

**GetData** - odczytuje dane

Próba połączenia z kimś wyglądała by następująco:

```
Winsock1.RemoteHost = "127.0.0.1"  
Winsock1.RemotePort = 5000  
Winsock1.Connect
```

Teraz ważniejsze właściwości naszej kontrolki czyli obsługa zdarzeń:

**Close()** - zamknięcie połączenia  
**Connect()** - utworzenie połączenia  
**ConnectionRequest(ByVal requestID As Long)** - ktoś z nami próbuje się połączyć, można takie połączenie zaakceptować przez polecenie **Accept** (**Winsock1.Accept requestID**), lub odrzucić poprzez zamknięcie połączenia  
**DataArrival(ByVal bytesTotal As Long)** - gdy dostaniemy dane od jakiegoś połączenia  
**Winsock1\_Error(...)** - gdy napotkamy błąd (np. przekroczony czas połączenia, albo gdy dokonamy próby wysłania danych gdy nie jesteśmy połączeni)  
**SendComplete()** - gdy dane przesyłane do nas zostały wysłane w całości  
**SendProgress(ByVal bytesSent As Long, ByVal bytesRemaining As Long)** - gdy dostaliśmy część danych

Przykładowe połączenie i wysłanie komunikatu powinno wyglądać mniej więcej tak:

```
Winsock1.RemoteHost = "127.0.0.1"  
Winsock1.RemotePort = 5000  
Winsock1.Connect  
  
Private Sub Winsock1_Connect()  
Winsock1.SendData "Witaj !" & Chr(13) & Chr(10)  
End Sub
```

Więcej o połączeniu można się dowiedzieć z lekcji Połączenie IRC oraz Połączenie FTP, tam wszystko powinno się wyjaśnić.

## 70. Połączenie IRC

<http://visual.basic.kaem.forall.pl/>

W tej lekcji spróbuję wyjaśnić na czym polega połączenie IRC przy pomocy kontrolki Winsock.

Pierwsze co nam będzie potrzebne to znaleźć jakiś działający serwer IRC-a, gdy już go mamy ustawiamy na formie jeden przycisk z tekstem 'Połącz' oraz kontrolke Winsock1, należy ustawić aby nasz program łączył się z wybranym przez nas serwerem irc po wciśnięciu przycisku, dodajemy więc:

```
Private Sub Command1_Click()  
Winsock1.Close  
Winsock1.RemoteHost = "irc.after-all.org"  
Winsock1.RemotePort = 6667  
Winsock1.Connect  
End Sub
```

Początkowo zostało użytek Winsock1.Close w przypadku gdyby połączenie było już

otwarte, inaczej kontrolka zwróci nam błąd że połączenie już istnieje. Coż jednak nam po samej próbie nawiązania połączenia skoro nawet nie wiemy czy zosaliśmy połączeni, dodajemy zdarzenie połączenia.

```
Private Sub Winsock1_Connect()  
MsgBox "Połączenie nawiązane"  
End Sub
```

Konieczne jest też dodanie obsługi błędu, lub chociaż wyświetlenie samego komunikatu błędu

```
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal  
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As  
Long, CancelDisplay As Boolean)  
MsgBox "Błąd połączenia " & Number & ": " & Description  
End Sub
```

Teraz bardzo ważna część, żeby móc się zalogować do serwera IRC należy podać kilka danych, dlatego trzeba dodać do zdarzenia gdy jesteśmy połączeni następujące dane:

```
Private Sub Winsock1_Connect()  
MsgBox "Połączenie nawiązane"  
Winsock1.SendData "NICK " & "testnick" & " " & Chr(13) & Chr(10)  
Winsock1.SendData "USER " & "testnick" & " " & Winsock1.LocalHostName & " X /0 :jakis  
opis" & Chr(13) & Chr(10)  
End Sub
```

Trzeba też sprawdzić czy jesteśmy poprawnie zalogowani i ogólnie zrobić obsługę komunikatów które dostajemy od serwera IRC, w tym celu trzeba na formie ustawić pole tekstowe (Text1) dosyć szerokie z ustawionym Multiline na True oraz suwakiem pionowym

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)  
Dim Informacje As String  
Winsock1.GetData Informacje' pobieramy informacje które otrzymaliśmy  
Text1.Text = Informacje & Chr(13) & Chr(10) & Text1.Text ' i dodajemy je w naszym  
polu tekstowym  
End Sub
```

Można już przetestować program, po udanym połączeniu nasze pole tekstowe powinno zapełnić się informacjami i komunikatami powitalnymi. Jesteśmy już połączeni z serwerem i zalogowani. Teraz trzeba zrobić obsługę komunikatów które dostajemy do serwera, ale nie będę tego opisywał bo to każdy powinien zrobić na swój sposób. Opisze jeszcze tylko wysyłanie podstawowych wiadomości. Dodajmy na formie kolejne pole tekstowe, tym razem zwykłe oraz kolejny przycisk. Tym polem użytkownik będzie mógł wysyłać informacje do serwera:

```
Private Sub Command2_Click()  
If Winsock1.State = 7 Then  
Winsock1.SendData Text2.Text & Chr(13) & Chr(10)  
Text2.Text = ""  
End If  
End Sub
```

```
Private Sub Text2_KeyPress(KeyAscii As Integer)  
If KeyAscii = 13 Then  
Command2_Click
```

End If  
End Sub

Pierwsze zdarzenie sprawdza czy kontrolka jest połączona z serwerem i jeśli tak to wysyła od nas informacje. Drugie zdarzenie sprawdza czy użytkownik wcisnął enter, jeśli tak wysyła informacje z pola tekstowego.

Podstawowymi poleceniami jakie możemy wysłać do serwera są:

**JOIN #kanal** - dołączenie do wybranego kanału, przy połączeniu dostajemy listę osób z kanału

**PART #kanal** - opuszczenie wybranego kanału rozmów

**QUIT** - wylogowanie się z serwera i zamknięcie połączenia

**MODE co kto** - zmiana atrybutów dla danej osoby jeżeli mamy takie uprawnienia

**PRIVMSG #kanal/nick :wiadomosc** - wysłanie wiadomości na kanał lub do wybranej osoby

Przykładowe wejście na jakiś kanał oraz napisanie na nim informacji wygląda tak:

JOIN #polska

PRIVMSG #polska :witam wszystkich !

Ostatnią bardzo ważną rzeczą jest wysłanie odpowiedzi PING/PONG, serwer co ustalony czas sprawdza czy połączenie jest poprawnie utrzymane dlatego wysyła do nas informacje PING + losowe dane, należy mu na to odpowiedzieć przez PONG + dane w talibcy, trzeba zmodyfikować otrzymywanie informacji w ten sposób:

Private Sub Winsock1\_DataArrival(ByVal bytesTotal As Long)

Dim Informacje As String

Winsock1.GetData Informacje

Text1.Text = Informacje & Chr(13) & Chr(10) & Text1.Text

If InStr(Informacje, "PING") = 1 Then

Winsock1.SendData "PONG " & Split(Informacje, " ")(1)

End If

End Sub

Opisałem najważniejsze rzeczy, teraz żeby sprawdzić stworzyć klienta IRC albo bota należy poświęcić dłuższą chwilę czasu, zrobić obsługę informacji których dostajemy, jest tego całkiem sporo, liczba osób na kanałach na których jesteśmy, ich atrybuty, temat na kanale, zmiana tematu, zmiana atrybutów, sprawdzanie kto na kanał dołącza kto wychodzi itp itd.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 71. Połączenie FTP

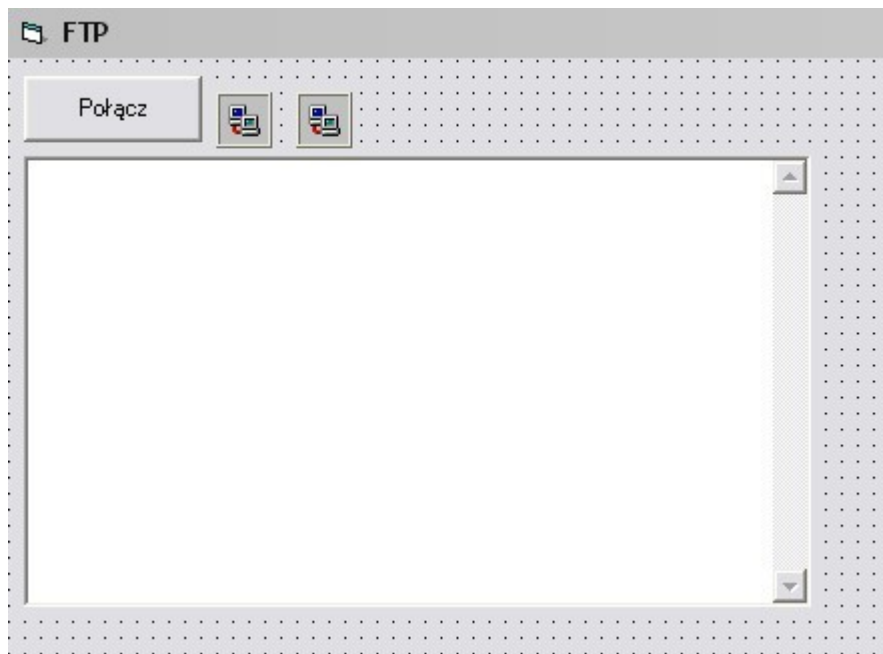
<http://visual.basic.kaem.forall.pl/>

Połączenie ftp jest trochę bardziej skomplikowane od połączenia IRC, przykładowy program do takiego połączenia zaczniemy budować podobnie jak połączenie z irc.

Na formie stawiamy narazie dwie kontrolki WinSock, przycisk i pole tekstowe z

ustawionym multiline oraz pionowym suwakiem. Połączenie będzie pokazywał na przykładzie łączenia się z ftp na moim domowym kompie, takie rozwiązanie jest najlepsze do testów. Wystarczy do tego np. trialowa wersja programu Serv-U Daemon. Tworzymy w nim testowe konto na porcie 21 i o loginie anonymous.

Ułożmy opisane wyżej elementy na formie w mniej więcej taki sposób:



Teraz trzeba dodać standardowe zdarzenia (jeżeli ktoś nie czytał lekcji o połączeniu IRC radzę najpierw ją przeczytać):

```
Private Sub Command1_Click()  
Winsock1.Close  
Winsock1.RemoteHost = "127.0.0.1"  
Winsock1.RemotePort = 21  
Winsock1.Connect  
End Sub
```

```
Private Sub Winsock1_Connect()  
Text1.Text = "Połączono"  
End Sub
```

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)  
Dim Informacje As String  
Winsock1.GetData Informacje  
Text1.Text = Informacje & Text1.Text  
End Sub
```

```
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal  
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As  
Long, CancelDisplay As Boolean)  
MsgBox "Błąd połączenia " & Number & ": " & Description  
End Sub
```

W tej chwili po wciśnięciu przycisku połącz powinniśmy dokonać poprawnego połączenia oraz otrzymać powitanie od naszego serwera:

220 Serv-U FTP Server v6.0 for WinSock ready...  
Połączono

Komunikacja między nami a serwerem jest dość prosta, my wysyłamy polecenie, serwer odpowiada komunikatem z odpowiednim numerem zdarzenia, np. 220 to komunikat powitania, 421 komunikat gdy za dużo użytkowników na ftp lub gdy stracimy połączenie z serwerem. Komunikaty, oprócz numerów, na różnych serwerach ftp mają różne postaci. Co więc robić żeby wiedzieć który komunikat do czego służy ? Najlepszym rozwiązaniem jest wprowadzenie zmiennej która będzie informowała nas co aktualnie na ftp się dzieje. Zadeklarujmy więc gdzieś zmienną:

#### Private Akcja As String

Oczywiście musimy także zadeklarować zmienną tymczasową która będzie pobierała numer z komunikatu który otrzymujemy.

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Dim Informacje As String
Winsock1.GetData Informacje
Dim Numer As String
Numer = Mid(Informacje, 1, 3)
Text1.Text = Informacje & Text1.Text
End Sub
```

Oraz przy dokonaniu połączenia ustawić wartość zmiennej Akcja na 0

```
Private Sub Winsock1_Connect()
Text1.Text = "Połączono"
Akcja = 0
End Sub
```

Teraz w zdarzeniu Winsock1\_DataArrival możemy już reagować na komunikaty i wysyłać przy tym odpowiednie polecenia. Będzie opisywał tylko prawidłowe komunikaty i tylko przy połączeniu anonymous.

Najpierw trzeba wysłać do serwera nazwę użytkownika i oczywiście zmienić numer Akcji (oczywiście robimy to w zdarzeniu DataArrival gdzieś przy dole:

```
If Akcja = 0 And Numer = "220" Then
Akcja = 1
Winsock1.SendData "USER anonymous" & Chr(13) & Chr(10)
End If
```

Teraz o ile wcześniej utworzyliśmy takie konto powinniśmy dostać komunikat:

331 User name okay, please send complete E-mail address as password.

Lub coś w tym rodzaju, zostało nam jeszcze wysłać hasło, dopisujemy nowe zdarzenie nad tym poprzednim:

```
If Akcja = 1 And Numer = "331" Then
Akcja = 2
Winsock1.SendData "PASS email@email.com" & Chr(13) & Chr(10)
End If
```

Jeżeli wszystko jest ok dostaniemy komunikat z numerem 230 że jesteśmy poprawnie zalogowani. Jeżeli nie dostaniemy komunikat z np. numerem 530 czyli złe hasło lub brak

anonimowych logować, na takie zdarzenie radzę sobie samemu zrobić reakcję. OK wracając do połączenia, jesteśmy już prawidłowo zalogowani na ftp. Teraz wypadało by ustawić katalog z którego chcemy pobrać listę plików. Ponieważ nie wiemy narazie jeszcze jakie katalogi są na ftp ustawmy narazie "/" jako nasz katalog. Katalog sprawdza się poleceniem CWD, należy więc napisać:

```
If Akcja = 2 And Numer = "230" Then
Akcja = 3
Winsock1.SendData "CWD" & Chr(13) & Chr(10)
End If
```

Jeżeli katalog istnieje otrzymamy '250 Directory changed to /', teraz należy ustawić go jako katalog domyślny z którego będziemy chcieli pobrać listę plików (wysyłamy polecenie PWD):

```
If Akcja = 3 And Numer = "250" Then
Akcja = 4
Winsock1.SendData "PWD" & Chr(13) & Chr(10)
End If
```

Do tej pory obsługa zdarzenia DataArrival powinna wyglądać mniej więcej tak:

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Dim Informacje As String
Winsock1.GetData Informacje
Dim Numer As String
Numer = Mid(Informacje, 1, 3)
Text1.Text = Informacje & Text1.Text
```

```
If Akcja = 3 And Numer = "250" Then
Akcja = 4
Winsock1.SendData "PWD" & Chr(13) & Chr(10)
End If
```

```
If Akcja = 2 And Numer = "230" Then
Akcja = 3
Winsock1.SendData "CWD /" & Chr(13) & Chr(10)
End If
```

```
If Akcja = 1 And Numer = "331" Then
Akcja = 2
Winsock1.SendData "PASS email@email.com" & Chr(13) & Chr(10)
End If
```

```
If Akcja = 0 And Numer = "220" Then
Akcja = 1
Winsock1.SendData "USER anonymous" & Chr(13) & Chr(10)
End If
End Sub
```

Należy zauważyć że akcje są ułożone malejąco, w innym przypadku mogły by się pomieszać warunki IF, można to też zrobić przez Case jeżeli komuś jest wygodniej.

Teraz należy ustawić typ przesyłania danych, mamy różne typy, TYPE A - ascii, I - binary, Z - kompresja. Do pobrania listy plików z ftp potrzebny będzie nam TYPE A:

```
If Akcja = 4 And Numer = "257" Then
```



```
Akcja = 5
Winsock1.SendData "TYPE A" & Chr(13) & Chr(10)
End If
```

Teraz trochę kłopotliwa rzecz, trzeba wybrać rodzaj połączenia. Jeżeli ftp ma zewnętrzny ip zawsze uda nam się z nim połączyć, jeżeli ip ma wewnętrzne ip i tylko port 21 jest na nie przekierowany połączymy się z nim tylko gdy my mamy zewnętrzne ip, naszczęście takich ftp jest mało. Założmy więc że łączymy się z ftp które ma zewnętrzne IP. W takim przypadku jeżeli nasz komputer ma wewnętrzne IP można się połączyć tylko przez tryb Pasywny, jeżeli nasz komputer ma zewnętrzne IP można się połączyć przez tryb Pasywny a także tryb Aktywny. Najkorzystniejsze będzie więc połączenie w trybie Pasywnym.

Omawiając te dwa tryby w bardziej zrozumiały sposób:

Połączenie aktywne: my nasłuchujemy na wybranym porcie, ftp przesyła nam dane

Połączenie pasywne: ftp nasłuchuje na wybranym porcie, łączymy się z nim i pobieramy dane

Aby ustawić połączenie pasywne należy wysłać polecenie PASV do naszego ftp:

```
If Akcja = 5 And Numer = "200" Then
Akcja = 6
Winsock1.SendData "PASV" & Chr(13) & Chr(10)
End If
```

Jeżeli do tej pory wszystko przebiega prawidłowo komunikaty które otrzymujemy powinny wyglądać mniej więcej tak:

```
227 Entering Passive Mode (127,0,0,1,7,85)
200 Type set to A.
257 "/" is current directory.
250 Directory changed to /
230 User logged in, proceed.
331 User name okay, please send complete E-mail address as password.
220 Serv-U FTP Server v6.0 for WinSock ready...
Połączono
```

Ostatni komunikat który otrzymamy po wysłaniu polecenia PASV '227 Entering Passive Mode (127,0,0,1,7,85)' zawiera w sobie informacje o numerze portu na którym musimy się połączyć aby pobrać listę plików i folderów z ftp. Oblicza się to z dwóch ostatnich cyfr. Żeby je wyciągnąć najlepiej pozbyć się nawiasu z prawej strony a następnie poleceniem Split podzielić cyfry na części. Nowe połączenie należy otworzyć drugą kontrolką.

```
If Akcja = 6 And Numer = "227" Then
Akcja = 7
Dim Port As Integer
Cyfra = Split(Mid(Informacje, 1, Len(Informacje) - 3), ",")
Port = Int(Cyfra(4)) * 256 + Int(Cyfra(5))
Winsock2.Connect Winsock1.RemoteHost, Port
End If
```

W powyższym kodzie żeby uciąć końcową spację z wyrażenia Informacje trzeba uciąć aż trzy znaki, chr(10) chr(13) i dopiero trzecią liczbą od prawej strony jest spacja. Teraz należało by jeszcze zrobić obsługę dla kontrolki Winsock2. Najpierw, jak już druga kontrolka się połączy należy wysłać polecenie do pierwszej kontrolki informujące o tym że chcemy pobrać listę plików, dla pewności można także zmienić numer akcji:

```
Private Sub Winsock2_Connect()
```

```

Akcja = 20
If Winsock1.State = 7 Then
Winsock1.SendData "LIST -al" & Chr(13) & Chr(10)
End If
End Sub

```

Po wysłaniu tej komendy do naszej kontrolki Winsock2 będzie przesyłana lista plików w formie RAW, ponieważ różnie ona wygląda na różnych ftp i nie zawsze wysyłana jest w jednym ciągu znaków najbezpieczniej było by ją zapisać do pliku a dopiero później z niej wyciągnąć odpowiednie dane, należy też zrobić obsługę błędów w razie gdyby coś poszło nie tak, np. nie otworzyło się nasze połączenie passive:

```

Private Sub Winsock2_DataArrival(ByVal bytesTotal As Long)
Dim ListaPlikow As String
Winsock2.GetData ListaPlikow
Open "lista.txt" For Append As #1
Print #1, ListaPlikow
Close #1
End Sub

```

```

Private Sub Winsock2_Error(ByVal Number As Integer, Description As String, ByVal
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As
Long, CancelDisplay As Boolean)
MsgBox "Błąd połączenia (lista plików) " & Number & ": " & Description
End Sub

```

Na przyszłość możemy także dodać usuwanie tej listy zanim zostanie stworzona. Inaczej plik nadpisywał by się w nieskończoność, zmieniamy w tym celu zdarzenie Winsock2\_Connect:

```

Private Sub Winsock2_Connect()
Akcja = 20
If Winsock1.State = 7 Then
Winsock1.SendData "LIST -al" & Chr(13) & Chr(10)
End If
On Error Resume Next
Kill "lista.txt"
End Sub

```

Gdy cała lista została już pobrana otrzymujemy od Winsock1 komunikat 226 Transfer complete, można już zamknąć połączenie Winsock2

```

If Akcja = 20 And Numer = "226" Then
Winsock2.Close
End If

```

Lista plików jest już pobrana i jest w pliku "lista.txt", powinna wyglądać mniej więcej tak:

```

drw-rw-rw- 1 user group 0 May 15 11:51 .
drw-rw-rw- 1 user group 0 May 15 11:51 ..
drw-rw-rw- 1 user group 0 Sep 24 14:14 incoming
drw-rw-rw- 1 user group 0 Sep 24 14:41 temp

```

Zapraszam do następnych lekcji o ftp gdzie będzie można się dowiedzieć jakie rodzaje listy plików są wysyłane przez FTP, jak wyciągnąć z nich potrzebne dane. Będą także opisane najważniejsze komendy połączenia ftp oraz pobieranie i wysyłanie plików, teraz zachęcam jeszcze do obejrzenia całości opisywanej tutaj.

Dla przeciwwiczenia wszystkiego radzę zrobić cały program jak tutaj opisywany tyle że użyć instrukcji Case zamiast ifów.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 72. FTP, komendy i listowanie

<http://visual.basic.kaem.forall.pl/>

W tej części kursu wyjaśnię zasady odczytywania plików i folderów z listy RAW, opiszę także najważniejsze komendy dotyczące FTP. Najpierw w źródle z poprzedniej lekcji należy dokonać kilku zmian. Przedewszystkim dobrze by było dodać dwa listboxy, jeden będzie na pliki, drugi na foldery. Oczywiście można to zrobić w inny sposób używając jakiegoś listviewa itp, ale pozostawimy narazie przy dwóch podstawowych ListBoxach. Dodatkowo pasowałoby trochę zmienić wysyłanie poleceń do kontrolki Winsock1. Dobrze byłoby zadeklarować osobną procedurę która przy okazji zapisuje do pola tekstowego Text1 informacje które wysyłamy:

```
Private Sub Slij(ByVal Tekst As String)
If Winsock1.State = 7 Then
Text1.Text = "--> " & Tekst & Chr(13) & Chr(10) & Text1.Text
Winsock1.SendData Tekst & Chr(13) & Chr(10)
Else
Text1.Text = "Nie można wysłać: " & Tekst & Chr(13) & Chr(10) & Text1.Text
End If
End Sub
```

Teraz trzeba pozmienić wszystkie informacje które wysyłamy, przykładowo zamienić:  
Winsock1.SendData "USER anonymous" & chr(13) & chr(10)  
na:  
Slij "USER anonymous"

Można też dokonać kilku poprawek:

Cyfra = Split(Replace(Informacje, "\"", "\",\"), "\",\"") - tak jest bezpieczniej

Winsock2.Close - pasowało by w razie pewności zamknąć połączenie  
Winsock2.Connect Winsock1.RemoteHost, Port

Jedną z ważniejszych rzeczy które należy zrobić jest zmiana zapisy listy do pliku. Zapisywanie listy po kawałku powoduje niechciane znaki entere po każdym zapisie, czasem lista dociera do nas w wielu częściach i przez to powstanie rozrzut w danych. Prawidłowy jest więc zapis który zapisuje wszystkie dane z listy do zmiennej, a dopiero pod koniec zapisuje dane do pliku. Można też dzięki temu pozbyć się usuwania pliku gdyż wystarczy że będzie otwarty tylko do zapisu.

Lista plików i folderów uzyskana od ftp może wyglądać tak:

```
09-20-05 08:01PM      <DIR>          folder
03-26-04 12:49PM      11452552 filmik z gry.wmv
```

Lub tak:

```
drw-rw-rw-  1 user   group      0 May 15 11:51 folder
-rw-rw-rw-  1 user   group  11452552 Sep 26 20:29 filmik z gry.wmv
```

Pierwsza najważniejsza rzecz to określenie z którym typem listy mamy doczynienia. Można to zrobić łatwo gdyż w 2 rodzaju pierwszym znakiem będzie zawsze albo "d" albo "-", w 1 rodzaju listy będzie to zawsze liczba. Teraz dobrze by było napisać osobną procedurę która będzie wyciągała foldery i pliki z różnych list. Dodajemy więc w zdarzeniu 'If Akcja = 20 And Numer = "226" Then' przejście do nowej procedury.

```
If Akcja = 20 And Numer = "226" Then
Winsock2.Close
Open "lista.txt" For Output As #1
Print #1, Lista
Close #1
Lista = ""
LadujListe
End If
```

Nasza procedura powinna wyglądać tak (troche rozbudowana ale łatwo ją zrozumieć):

```
Private Sub LadujListe()
Dim Znak As String
Dim Linia As String
Dim Sposob As Byte

Dim Nazwa As String
Dim Rozmiar As String
Dim C As Integer

List1.Clear
List2.Clear
Open "lista.txt" For Input As #1 ' otwieramy plik z listą plików

Do While (EOF(1) = False) ' wczytujemy linijki dopókinie natrafimy na koniec pliku
Line Input #1, Linia

    If Linia <> "" Then

        Znak = Mid(Linia, 1, 1) ' sprawdzamy 1 znak w linijce
        Sposob = 2 ' automatycznie przydzielamy Sposob jako 2
        If InStr(1, Linia, "<DIR>") > 0 Then Sposob = 3 ' gdy znajdziemy <DIR> jako 3
        If Znak = "d" Then Sposob = 0 ' gdy znak = d sposob = 0
        If Znak = "-" Then Sposob = 1 ' gdy -, = 1

        If Sposob = 0 Or Sposob = 1 Then
            C = Szukaj(Linia, 12) ' pozbywamy sie wartosci poczatkowych z listy
            Linia = Mid(Linia, C)
            C = Szukaj(Linia, 8)
            Linia = Mid(Linia, C)
            C = Szukaj(Linia, 6)
            Linia = Mid(Linia, C)
            Rozmiar = Mid(Linia, 1, InStr(1, Linia, " ")) ' wycinamy rozmiar
            Nazwa = Mid(Linia, InStr(1, Linia, " ") + 14) ' i nazwe
            If Sposob = 0 Then List1.AddItem Nazwa ' dodajemy folder do list1
            If Sposob = 1 Then List2.AddItem Nazwa ' dodajemy plik do list2
        End If
    End If
End While
```

```

If Sposob = 2 Then
C = Szukaj(Linia, 18) ' pozbywamy sie wartosci poczatkowych z listy
Linia = Mid(Linia, C)
Rozmiar = Mid(Linia, 1, InStr(1, Linia, " "))
Nazwa = Mid(Linia, InStr(1, Linia, " ") + 1)
List2.AddItem Nazwa ' dodajemy plik do list2
End If

If Sposob = 3 Then
Nazwa = Mid(Linia, 40)
List1.AddItem Nazwa ' dodajemy plik do list1
End If

End If

Loop

Close #1
End Sub

Private Function Szukaj(Linia As String, Start As Byte) As Integer
Dim A As Byte
Dim Dlug As Integer
Dim Znak As String
A = Start
Dlug = Len(Linia)

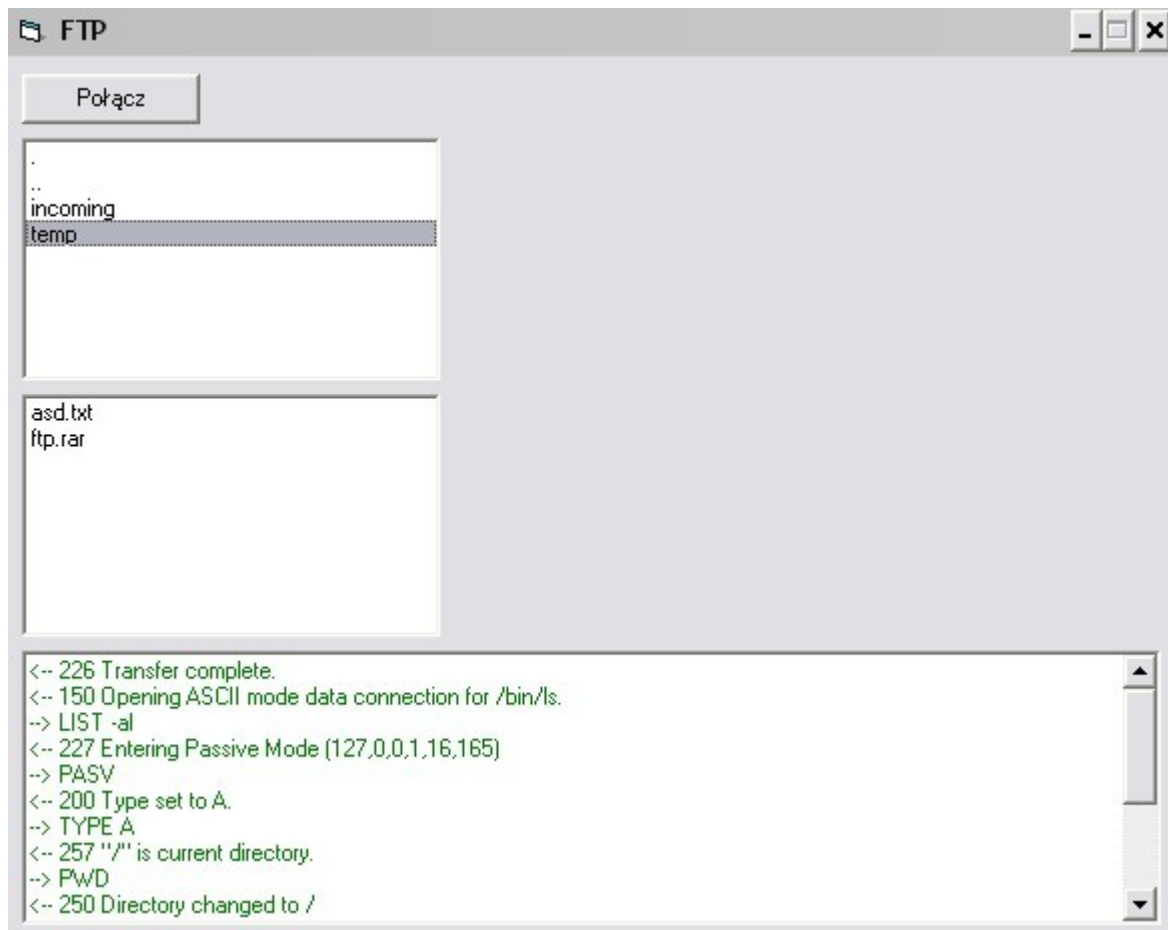
Znak = " "

Do While A < Dlug And Znak = " "
Znak = Mid(Linia, A, 1)
A = A + 1
Loop

Szukaj = A - 1
End Function

```

Ok, teraz pliki i foldery dodają się odpowiednio do wybranych pól List:



Przydało by się dać użytkownikowi możliwość wchodzenia do folderów. Będzie nam potrzebna zmienna przechowująca ścieżkę w której aktualnie jesteśmy + obsługa zdarzenia DbClick w liście plików. Trzeba zadeklarować zmienną:

**Private Sciezka As String**

oraz zmodyfikować odczytywanie komunikatu z PWD i wyciągnąć z niego aktualną ścieżkę:

```
If Akcja = 4 And Numer = "257" Then
Akcja = 5
SciezkaTemp = Split(Informacje, Chr(34))
Sciezka = SciezkaTemp(1) & If(Len(SciezkaTemp(1)) > 1, "/", "")
Slij "TYPE A"
End If
```

Akcje listowania plików i folderów z wybranego folderu z ftp już mamy ( od Akcji 3 do 6 ), więc należy ustawić Akcje na 3 po kliknięciu w List1:

**Private Sub List1\_DbClick()**

```
If List1.ListIndex > -1 And Winsock1.State = 7 Then ' gdy jesteśmy połączeni z ftp i jest
zaznaczony folder z List1 to
Akcja = 3 'Ustawiamy akcje na 3
Slij "CWD " & Sciezka & List1.List(List1.ListIndex) & "/" ' i wysyłamy komendę przejścia
do nowego katalogu
End If
```

## End Sub

Można już swobodnie poruszać się po naszym ftp. Teraz wystarczy stworzyć obsługę podstawowych poleceń, takich jak:

**CDUP** - przejście o jeden folder do góry

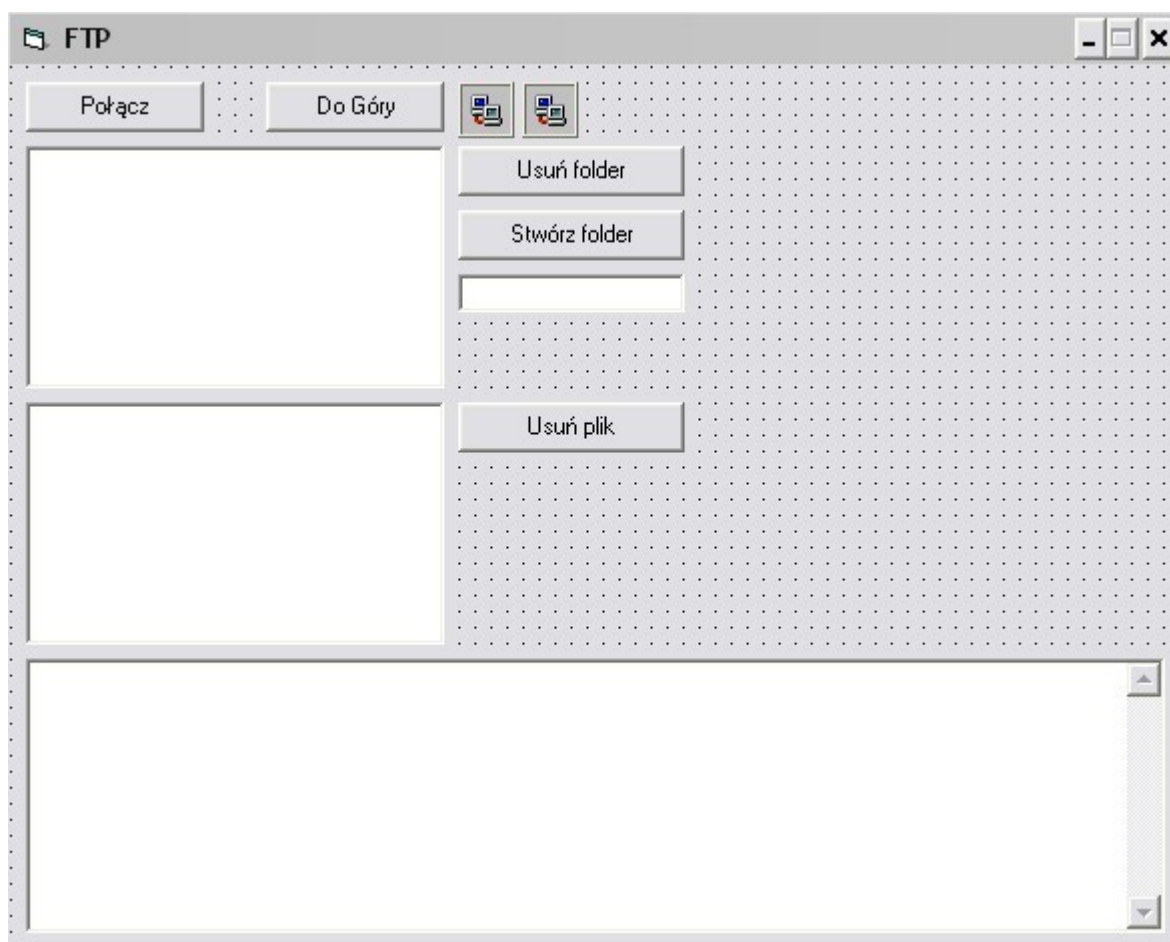
**DELE** - usunięcie pliku

**RMD** - usunięcie folderu (tylko gdy jest pusty)

**RNFR / RNTD** - zmiana nazwy folderu / pliku + przeniesienie folderu pliku

**MKD** - stworzenie folderu

Można to zrobić w postaci rozwijanego menu gdy klikniemy prawym przyciskiem myszy na List1 lub List2, jednak prościej będzie to zrobić na zwykłych przyciskach. Ustawmy na formie 4 przyciski i jedno pole tekstowe, powinno to wyglądać tak jak na obrazku



Teraz wystarczy ustawić odpowiednie zdarzenia dla przycisków. Najpierw Do Góry czyli CDUP, w tym zdarzeniu należy przesłać do FTP polecenie CDUP a następnie pobrać listę plików, wystarczy więc zrobić to samo co przy wchodzeniu do folderów:

```
Private Sub Command7_Click()
```

```
    If Winsock1.State = 7 Then
```

```
        Akcja = 3
```

```
        Slij "CDUP"
```

```
    End If
```

End Sub

Usuwanie folderu to zwykłe przesłanie 'RMD nazwa\_folderu' i odświeżenie listy plików:

```
Private Sub Command2_Click()
```

```
If List1.ListIndex > -1 And Winsock1.State = 7 Then
```

```
Dim CzyChcesz As Byte
```

```
CzyChcesz = MsgBox("Czy napewno chcesz usunąć folder: " & List1.List(List1.ListIndex),  
vbYesNo, "Potwierdzenie")
```

```
    If CzyChcesz = 6 Then
```

```
        Akcja = 3
```

```
        Slij "RMD " & List1.List(List1.ListIndex)
```

```
    End If
```

```
End If
```

```
End Sub
```

Stworzenie folderu to także rzecz bardzo prosta:

```
Private Sub Command3_Click()
```

```
If Winsock1.State = 7 And Text2.Text <> "" Then
```

```
Akcja = 3
```

```
Slij "MKD " & Text2.Text
```

```
End If
```

```
End Sub
```

Jednak przy tym polecenie powstaje mały problem, jeśli uda nam się stworzyć folder otrzymamy komunikat z numerem 257 a nasza Akcja = 3 działa tylko z komunikatem 250, należy dokonać małej zmiany, gdyż inaczej lista by się nie odświeżała:

```
If Akcja = 3 And (Numer = "250" Or Numer = "257") Then
```

```
Akcja = 4
```

```
Slij "PWD"
```

```
End If
```

Usuwanie pliku, jeżeli się powiedzie zwraca komunikat 250, wystarczy więc napisać prosty kod:

```
Private Sub Command5_Click()
```

```
If Winsock1.State = 7 And List2.ListIndex > -1 Then
```

```
Akcja = 3
```

```
Slij "DELE " & List2.List(List2.ListIndex)
```

```
End If
```

```
End Sub
```

Została jeszcze zmiana nazwy pliku, przeniesienie pliku, zmiana nazwy folderu i przeniesienie folderu. Do tych zdarzeń będzie trzeba napisać osobne zdarzenia akcji. Nie będę tego narazie opisywał, tylko wyjaśnię jak działają te dwa polecenia:

```
RNFR /folder/folder2/jakisfolder/
```



RNTO /folder/folder2/nowanazwa/  
- zmieni tylko nazwe folderu

RNFR /folder/folder2/jakisfolder/  
RNTO /folder/jakisfolder/  
- przeniesie folder w inne miejsce

RNFR /folder/folder2/jakisfolder/  
RNFR /nowanazwa/  
- przeniesie folder i zmieni jego nazwe.

Z plikami jest analogicznie.

Co jeszcze wypadalo by zrobic w naszym programie ? Można by ustawic tak zwane Keep Alive, czyli utrzymywanie połączenia z FTP poprzez wysyłanie komendy NOOP. Taką komendę wypadalo by wysłać po 30 sekundach braku ruchu ze strony użytkownika. Umieścmy więc na formie Timer z Interval = 1000 i zadelkarujmy zmienną Czas.

```
Private Sub Timer1_Timer()  
If Winsock1.State = 7 Then  
Czas = Czas + 1  
If Czas = 30 Then Slij "NOOP"  
End If  
End Sub
```

Zmienna czas musi się oczywiście zerować gdy prześlemy jakąś komendę do FTP:

```
Private Sub Slij(ByVal Tekst As String)  
Czas = 0  
If Winsock1.State = 7 Then  
Text1.Text = "--> " & Tekst & Chr(13) & Chr(10) & Text1.Text  
Winsock1.SendData Tekst & Chr(13) & Chr(10)  
Else  
Text1.Text = "Nie można wysłać: " & Tekst & Chr(13) & Chr(10) & Text1.Text  
End If  
End Sub
```

Należy zastanowić się jeszcze nad kilkoma sprawami i poprawić trochę cały program. Otóż może się zdarzyć że ftp wyśle nam dwa polecenia w jednej wiadomości, my natomiast jako numer wiadomości odczytujemy tylko pierwsze trzy cyfry. Przykładowo można czasem dostać takie dwie informacje w jednym ciągu danych:

```
<-- 150 Opening ASCII mode data connection for /bin/ls.  
226 Transfer complete.
```

Wtedy nasz program rozpozna tylko numer 150, co zrobić żeby to poprawić ? Należało by przy samym dole zdarzenia Winsock1\_DataArrival sprawdzić ile komunikatów dostaliśmy i jeżeli więcej niż jeden to wrócić do początku sprawdzania:

```
Dim Ile As Integer  
Info = Split(Informacje, Chr(13) & Chr(10))  
Ile = UBound(Info)  
If Ile > 1 Then  
Informacje = Mid(Informacje, InStr(Informacje, Chr(13) & Chr(10)) + 2)  
Numer = Mid(Informacje, 1, 3)  
GoTo OdNowa  
End If
```

Oczywiście OdNowa: deklarujemy zaraz przy początku zdarzenia:

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Dim Informacje As String
Winsock1.GetData Informacje
Dim Numer As String
Numer = Mid(Informacje, 1, 3)
Text1.Text = "<-- " & Informacje & Text1.Text
```

OdNowa:

...

Ostatnią rzeczą którą można poprawić program jest wstawienie małego przedziału czasowego po którym wysyła się komunikat do FTP, teraz gdy dostaniemy jakąś wiadomość od FTP program od razu wysyła na nią odpowiedź, można by zrobić między komunikatem a odpowiedzią 100 milisekund przerwy. Położymy na formie nowy Timer z Enabled = False i Interval = 100, zadeklarujemy też zmienną Wiadomosc jako String. Wszystko ma działać tak, aby komenda Slij wywoływała nowy timer który po 100 milisekundach będzie wysyłał nasz komunikat, zmieniamy całkowicie procedure Slij:

```
Private Sub Slij(ByVal Tekst As String)
Czas = 0
Wiadomosc = Tekst & Chr(13) & Chr(10)
Timer2.Enabled = True
End Sub
```

Oraz dodajemy obsługę timera:

```
Private Sub Timer2_Timer()
Timer2.Enabled = False
Text1.Text = "--> " & Wiadomosc & Text1.Text
If Winsock1.State = 7 Then
Winsock1.SendData Wiadomosc
Else
Text1.Text = "Nie można wysłać: " & Wiadomosc & Text1.Text
End If
End Sub
```

Tej lekcji już koniec, w następnej opisz transfer pliku z ftp na hdd.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 73. FTP, transfer danych

<http://visual.basic.kaem.forall.pl/>

Wszystko będziemy robili na formie z poprzedniej lekcji.

Do transferu danych będzie nam potrzebne kolejna kopia kontrolki WinSock, wstawiamy więc na naszą formę Winsock3. Do przesyłania plików trzeba wykorzystać przesyłanie w trybie binarnym (TYPE I). Pod naszym przyciskiem 'Usuń plik' stawiamy nowy przycisk

'Pobierz plik'. Komendą do pobierania plików jest RETR, do wysyłania STOR. Komendą pobierania pliku od wybranego momentu jest REST (ustawiamy np REST 1024, potem dajemy RETR i pobierany plik zaczyna się od 1024 bajta). Ponieważ będziemy musieli użyć nowego typu pobierania należy stworzyć specjalnie do tego nowe akcje. Najpierw zdarzenie które ma nastąpić po kliknięciu na przycisk pobierania:

```
Private Sub Command4_Click()  
If List2.ListIndex > -1 Then  
Akcja = 50  
Slij "TYPE I"  
End If  
End Sub
```

Ustawienie odpowiedzi na tą akcje i ustawienie trybu passive:

```
If Akcja = 50 And Numer = "200" Then  
Akcja = 51  
Slij "PASV"  
End If
```

Wyciągnięcie portu z passive i otwarcie nowego połączenia:

```
If Akcja = 51 And Numer = "227" Then  
Akcja = 52  
Cyfra = Split(Replace(Informacje, " ", ","), ",")  
Port = Int(Cyfra(4)) * 256 + Int(Cyfra(5))  
Winsock3.Close  
Winsock3.Connect Winsock1.RemoteHost, Port  
End If
```

Teraz podobnie do pobierania listy plików należy zrobić pobieranie pliku. Schemat jest ten sam, dostajemy dane i zapisujemy je do pliku.

```
Private Sub Winsock3_Connect() ' gdy się połączymy  
Akcja = 53  
Slij "RETR " & List2.List(List2.ListIndex) ' wysyłamy polecenie pobrania pliku  
End Sub
```

```
Private Sub Winsock3_DataArrival(ByVal bytesTotal As Long) ' pobieranie danych  
Czas = 0  
Dim Dane As String  
Winsock3.GetData Dane  
Open List2.List(List2.ListIndex) For Append As #2  
Print #2, Dane;  
Close #2  
End Sub
```

```
Private Sub Winsock3_Error(ByVal Number As Integer, Description As String, ByVal  
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As  
Long, CancelDisplay As Boolean)  
MsgBox "Błąd połączenia (pobieranie) " & Number & ": " & Description  
End Sub
```

Różnicami są: wyzerowanie czasu przy otrzymaniu danych, inaczej przy dużych plikach które pobierają się więcej niż 30 sekund program próbował by wysłać do FTP polecenie NOOP. Można by też wyłączyć Timer liczący ten czas i gdy w akcji 53 dostaniemy 226 czyli transfer complete znów włączyć timer. Ciekawą rzeczą na którą natrafiłem jest

wykorzystanie znaku ; (druga pogrubiona linijka), powoduje on że polecenie Print nie stawia na końcu dopisywanej linijki znaków chr(13) i chr(10) które oczywiście zepsuły by nasz pobierany plik (nie może być żadnych nowych bajtów, każdy bajt musi być taki sam).

Wysyłanie pliku ma trochę inną postać. Musimy otworzyć połączenie Pasywne, otworzyć nasz plik z dysku i wysłać go w kawałkach (polecam 8192 bajty czyli 8 kilobajtów). Gdy wyślemy taki pakiet dostaniemy potwierdzenie w zdarzeniu SendComplete i będzie można wysłać następny pakiet. Tak w kółko aż nie wyślemy całego pliku. Wtedy należy zamknąć połączenie. Nie będę już opisywał tutaj dokładnie kodu gdyż każdy powinien zrobić to samemu. Kod programu jest tylko szkieletem programu do FTP, więc jeśli ktoś chciałby się zabrać za takie coś radzę napisać wszystko od nowa i uwzględnić wiele rzeczy których nie uwzględniłem.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 74. FTP, listowanie<sup>3</sup>

<http://visual.basic.kaem.forall.pl/>

Opisze tutaj kilka sprostowań odnośnie wybierania nazwy plików, folderów oraz rozmiarów z listy plików pobieranej z ftp.

We wcześniejszej lekcji listowanie plików zostało opisane na przykładzie dwóch formatów danych dostarczanych przez różne serwery ftp. O ile odnośnie drugiego formatu wszystko działa poprawnie, w pierwszym czasem mogą nastąpić pewnie nieprawidłowości, zastosowanie poniższych użytych wcześniej funkcji:

```
If Sposob = 0 Or Sposob = 1 Then
C = Szukaj(Linia, 12) ' pozbywamy sie wartosci poczatkowych z listy
Linia = Mid(Linia, C)
C = Szukaj(Linia, 8)
Linia = Mid(Linia, C)
C = Szukaj(Linia, 6)
Linia = Mid(Linia, C)
Rozmiar = Mid(Linia, 1, InStr(1, Linia, " ")) ' wycinamy rozmiar
Nazwa = Mid(Linia, InStr(1, Linia, " ") + 14) ' i nazwe
...
End If
```

Nie jest do końca prawidłowe. W tym przykładzie szukamy jakiegokolwiek znaku który występuje po odstępach między wyrazami w postaci spacji. Jednak czasem odstępy te, oraz wyrazy ulegają zmianą, dlatego należy wprowadzić dodatkowo wyszukiwanie pierwszej spacji następującej po danym wyrazie.

```
Private Function SzukajSpacji(Linia As String, Start As Byte) As Integer
Dim a As Byte
Dim Dlug As Integer
Dim Znak As String
a = Start
Dlug = Len(Linia)
Znak = "X"
```

```

Do While a < Dlug And Znak <> " "
Znak = Mid(Linia, a, 1)
a = a + 1
Loop

```

```

SzukajSpacji = a - 1
End Function

```

Prawidłowy zapis po wprowadzeniu tej funkcji powinien wyglądać mniej więcej tak:

```

If Sposob = 0 Or Sposob = 1 Then

Dim C As Long
Dim D As Long

D = SzukajSpacji(Linia, 1)
C = Szukaj(Linia, D + 1)
D = SzukajSpacji(Linia, C + 3)
C = Szukaj(Linia, D + 1)
D = SzukajSpacji(Linia, C + 1)
C = Szukaj(Linia, D + 1)

Rozmiar = Mid(Linia, C, InStr(C, Linia, " ") - C)
Nazwa = Mid(Linia, InStr(C, Linia, " ") + 14)

...

End If

```

Ważne jest także żeby nie obcinać nazwy folderu ze zbędnych spacji używając polecenia Trim. Z reguły ani folder ani plik nie może mieć spacji na początku lub końcu swojej nazwy, jednak w praktyce jest zupełnie inaczej. Błędy windows pozwalają tworzyć takie spacje, w systemach typu Unix także zazwyczaj można tworzyć początkowe i końcowe spacje. Programy które je ucinają (np. FlashFXP) nie pozwalają na wejście do takich folderów. Przykładowo jeżeli folder nazywa się ' folder' to FlashFXP wyświetli go jako 'folder', gdy będziemy chcieli do niego wejść, otrzymamy błąd.

## 75. Połączenie HTTP

<http://visual.basic.kaem.forall.pl/>

Połączenie http jest znacznie prostsze w obsłudze niż ftp. Polega na wysłaniu do serwera http zapytania o plik oraz podania kilku danych. Przykładowo aby pobrać stronę <http://visual.basic.kaem.forall.pl/index.php> należy najpierw połączyć się z hostem visual.basic.kaem.forall.pl a następnie 'poprosić' o plik index.php.

Łączymy się standardowo:

```

Winsock1.Close
Winsock1.Connect "visual.basic.kaem.forall.pl", 80

```

Gdy już uda nam się połączyć należy wysłać odpowiednie informacje, są to między innymi:

- GET /nazwapliku rodzaj\_połączenia
- Host: nazwa\_hosta

- Accept: akceptowanie\_plikow
- Referer: strona\_referująca
- User-Agent: nazwa\_przeglądarki
- Pragma: rodzaj\_cachu
- Cache-Control: kontrolowanie\_cachu
- Connection: info\_o\_połączeniu

rodzaj połączenia obecnie używany to HTTP/1.1

host to nazwa naszego hosta

accept lepiej ustawić domyślnie na \*/\*

referer to stron odsyłająca, najlepiej zostawić puste lub wpisać nazwę naszego hosta razem z http

user-agent to nazwa przeglądarki, można wpisać coś od siebie lub podszyć się pod jakąś przeglądarkę

pragma to informacje o cachu strony, najlepiej tego nie używać gdyż wtedy zawsze będzie pobierało najnowszą wersję pliku

Wysłanie informacji o pobranie pliku 'index.php' wyglądało by więc następująco:

```
Private Sub Winsock1_Connect()
If Winsock1.State = 7 Then
Winsock1.SendData "GET /index.php HTTP/1.1" & Chr(13) & Chr(10)
Winsock1.SendData "Host: visual.basic.kaem.forall.pl" & Chr(13) & Chr(10)
Winsock1.SendData "Accept: */*" & Chr(13) & Chr(10)
Winsock1.SendData "Referer: http://visual.basic.kaem.forall.pl" & Chr(13) & Chr(10)
Winsock1.SendData "User-Agent: Mozilla/4.0 (compatible; MSIE 5.00; Windows 98)" & Chr(13) & Chr(10)
Winsock1.SendData "Pragma: no-cache" & Chr(13) & Chr(10)
Winsock1.SendData "Cache-Control: no-cache" & Chr(13) & Chr(10)
Winsock1.SendData "Connection: close" & Chr(13) & Chr(10) & Chr(13) & Chr(10)
End If
End Sub
```

Po wysłaniu tych informacji serwer z którym się łączymy powinien nam zwrócić pożądaną stronę lub, gdy coś pójdzie nie tak stronę błędu.

Serwer zwraca także trochę podstawowych informacji o stronie i serwerze z którego jest pobierana, w naszym przypadku będą to następujące informacje:

```
HTTP/1.1 200 OK
Date: Thu, 06 Oct 2005 18:11:39 GMT
Server: Apache/2.0.54 (Unix)
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
```

200 OK oznacza dostarczenie poprawnej strony www / pliku, z ważniejszych błędów na które można się natknąć to 400 Bad Request czyli wysłanie złego zapytania o pobieraną stronę/plik.

Dostajemy także obecną datę na serwerze, informacje o serwerze i inne narazie nie potrzebne nam dane.

Po otrzymaniu tych danych zostaje wysyłany podwójny znak enteru [Chr(13) & Chr(10)] a bezpośrednio po nim treść naszej strony.

Oczywiście odbierane informacje wypadało by zapisać do jakiegoś pliku:

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
If Winsock1.State = 7 Then
Dim Dane As String
Winsock1.GetData Dane
Open "index.txt" For Append As #1
Print #1, Dane;
Close #1
End If
End Sub
```

W ten sposób zapiszemy wszystkie dane do pliku (razem z informacjami o stronie i serwerze), plik będzie zaczynał się mniej więcej tak:

```
HTTP/1.1 200 OK
Date: Thu, 06 Oct 2005 18:24:05 GMT
Server: Apache/2.0.54 (Unix)
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
```

```
204f
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">...
```

Oczywiście żeby wszystko było poprawnie należy oddzielić dane od serwera, nagłówek pliku (w przykładzie jest to np. 204f) oraz właściwą część pliku.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 76. Algorytm: NWD

<http://visual.basic.kaem.forall.pl/>

### NWD (Największy Wspólny Dzielnik)

Algorytm ten nazywany inaczej algorytmem Euklidesa ma za zadanie znaleźć największy wspólny dzielnik dla dwóch dowolnych liczb.

Zasady działania są proste, program bierze 2 liczby i dzieli liczbę większą przez mniejszą obliczając resztę z dzielenia. Jeżeli reszta jest równa 0 algorytm przerywa działanie, w innym wypadku następuje podmienienie liczb (większa > mniejsza, mniejsza > reszta z dzielenia) i ponowne sprawdzenie dzielenia.

Przykład dla dwóch liczb:

A = 6025

B = 505

Reszta z 6025 : 505 = 25

podmieniamy 6025 z 25, wtedy

A = B = 505  
B = reszta = 25

Reszta z 505 : 25 = 5

podmieniamy 505 z 5, wtedy

A = B = 25  
B = reszta = 5

Reszta z 25 : 5 = 0, czyli  
NWD = B = 5

Zapis tego działania jako kodu Visual Basic jest bardzo prosty, można sobie stworzyć na przykład funkcję:

Function NWD(A As Long, B As Long)

Do While B <> 0  
Temp = A Mod B  
A = B  
B = Temp  
Loop

NWD = A

End Function

Użycie funkcji wygląda tak:

Dzielnik = NWD(Liczba1,Liczba2)

Należy pamiętać aby Liczba1 była większa od Liczby2 chyba że zmodyfikuje algorytm tak aby sam wybierał większą liczbę.

Function NWD(A As Long, B As Long)

If B > A Then  
Dim Temp As Long  
Temp = B  
B = A  
A = Temp  
End If

Do While B <> 0  
Temp = A Mod B  
A = B  
B = Temp  
Loop

NWD = A

End Function

Działanie algorytmu jest oparte na pętli która oblicza kolejno resztę z dzielenia aż do momentu gdy reszta będzie równa zero, gdy reszta jest różna od 0 następują zmiana liczb. Reszta z dzielenia pobieramy stosując Mod ( modulo ).



# 77. Algorytm: Sortowanie Bąbelkowe

<http://visual.basic.kaem.forall.pl/>

Ten rodzaj sortowania jest jednym z najmniej wydajnych sortowań, jednak zasada działania jest dość prosta i pomoże zrozumieć inne metody.

Polega to na sprawdzaniu sąsiadujących elementów z tablicy a następnie jeżeli element z lewej strony jest większy (lub gdy chcemy posortować odwrotnie mniejszy) niż element z prawej strony następuje zamiana elementów.

Sama zamiana występuje w postaci takiego schematu:

```
If ElementLewy > ElementPrawy Then  
Temp = ElementLewy  
ElementLewy = ElementPrawy  
ElementPrawy = Temp  
End If
```

Żeby sprawdzić wszystkie sąsiadujące ze sobą elementy z tablicy którą chcemy posortować należało by więc użyć pętli powtarzającej się X razy (X jest liczbą elementów z tablicy odjąć 1, w innym przypadku sprawdzalibyśmy ostatni element tablicy z elementem nieistniejącym).

Samo jedno sprawdzenie elementów z tablicy nie jest wystarczające, dla przykładu posortujmy liczby używając schematu:

```
Liczby = 1,17,9,4  
Sprawdzamy 1 i 2 element: Jeżeli 1 > 17 - nie spełniony warunek  
Sprawdzamy 2 i 3 element: Jeżeli 17 > 9 - warunek spełniony, podmieniamy liczby  
Liczby = 1,9,17,4  
Sprawdzamy 3 - 4 element: Jeżeli 17 > 4 - warunek spełniony, podmieniamy liczby  
Liczby = 1,9,4,17
```

Odrzu można zauważyć że sortowanie nie jest zakończone, co więc należy zrobić ?  
Dokonać sortowania X - 1 razy, w ten sposób dokonamy 3 sortowań i w każdym z nich dokonamy 3 porównań.

**1 sortowanie:**

```
Liczby = 1,17,9,4  
Sprawdzamy 1 i 2 element: Jeżeli 1 > 17 - nie spełniony warunek  
Sprawdzamy 2 i 3 element: Jeżeli 17 > 9 - warunek spełniony, podmieniamy liczby  
Liczby = 1,9,17,4  
Sprawdzamy 3 - 4 element: Jeżeli 17 > 4 - warunek spełniony, podmieniamy liczby  
Liczby = 1,9,4,17
```

**2 sortowanie:**

```
Liczby = 1,9,4,17  
Sprawdzamy 1 i 2 element: Jeżeli 1 > 9 - nie spełniony warunek  
Sprawdzamy 2 i 3 element: Jeżeli 9 > 4 - warunek spełniony, podmieniamy liczby  
Liczby = 1,4,9,17  
Sprawdzamy 3 i 4 element: Jeżeli 9 > 17 - nie spełniony warunek
```

**3 sortowanie:**

```
Sprawdzamy 1 i 2 element: Jeżeli 1 > 4 - nie spełniony warunek  
Sprawdzamy 2 i 3 element: Jeżeli 4 > 9 - nie spełniony warunek  
Sprawdzamy 3 i 4 element: Jeżeli 9 > 17 - nie spełniony warunek
```

Trzecie sortowanie nic w naszych liczbach nie zmienia, jednak jest obowiązkowe gdyż nie wiadomo jak ułożone są elementy w tablicy, w przypadku gdybyśmy chcieli posortować liczby 14,5,3,1 trzecie sortowanie było by potrzebne do prawidłowego posortowania tablicy.

Przez taki układ sortować Sortowanie Bąbelkowe nie jest zbyt wydajne, następuje

$(n - 1) * (n - 1)$  porównań liczb, ( po pomnożeniu uzyskamy  $n$  do kwadratu -  $2n + 1$ , dla posortowania 100 elementów algorytm musi dokonać 100 do kwadratu odjąć 2 razy  $100 + 1$  porównań ( 9801 porównań ! )).

Można trochę usprawnić ten algorytm jednak najpierw pokaże zapis tego do czego narazie doszliśmy.

```
For J = 0 To 49 Step 1
  For I = 0 To 49 Step 1
    If Tablica(I) > Tablica(I + 1) Then
      Temp = Tablica(I)
      Tablica(I) = Tablica(I + 1)
      Tablica(I + 1) = Temp
    End If
  Next I
Next J
```

W powyższym kodzie sprawdzamy tablicę 51 elementową (od 0 do 50). Pętla J to ilość sortowań, Pętla I Sprawdza pokolei wszystkie pary elementów i wykonuje się J razy.

Algorytm ten da się napisać wprowadzając jedną poprawkę (mimo tego sortowanie i tak będzie bardzo wolne), mianowicie można zauważyć że przy każdym sortowaniu elementy od końca są już ułożone. Po 1 sortowaniu ostatni element będzie już ułożony prawidłowo, po 2 sortowaniu 2 ostatnie elementy będą już prawidłowe, po 3 sortowaniu 3 elementy i tak dalej. Dlatego porównywanie elementów można stopniowo skracać. Porównując za 1 razem np. 50 elementów, potem 49,48,47 itp. Zapis wygląda tak:

```
For J = 0 To 49 Step 1
  For I = 0 To 49 - J Step 1
    If Tablica(I) > Tablica(I + 1) Then
      Temp = Tablica(I)
      Tablica(I) = Tablica(I + 1)
      Tablica(I + 1) = Temp
    End If
  Next I
Next J
```

Od liczby porównań odejmuje J (pogrubiony wyraz). Sortowane jest zrobione dla  $n$  elementów, porównania dla  $n-1$  elementów, dlatego ogólny zarys algorytmu dla  $N$  elementów wygląda tak:

```
For J = 0 To N - 1 Step 1
  For I = 0 To N - 1 - J Step 1
    If Tablica(I) > Tablica(I + 1) Then
      Temp = Tablica(I)
      Tablica(I) = Tablica(I + 1)
      Tablica(I + 1) = Temp
    End If
  Next I
Next J
```

Obliczmy teraz wydajność algorytmu po zmianie. Teraz stopniowo zmniejszamy liczbę porównań elementów, za pierwszym razem o 0, potem o 1,2,3,4,5 aż do  $N - 2$  (dlatego że od  $N - 1$  odejmujemy jeszcze jeden element). Dlatego od poprzedniego zapisu wystarczy odjąć sumę elementów ciągu  $0, 1, 2, 3, \dots, N - 2$

$$(n - 1) * (n - 1) - 0, 1, 2, 3, 4, \dots, N - 2$$

Sumę wyrazów ciągu można obliczyć z wzoru,  $S_n = n * (a_n - a_0) / 2$ ,  $a_0$  jest 1 elementem ciągu,  $a_n$  ostatnim, czyli podstawiając nasz ciąg:

$$0, 1, 2, \dots, N-1 = N * (N-1-0) / 2 = N * (N - 2) / 2$$

Podstawiając sumę ciągu otrzymujemy:

$(n-1)(n-1) - n(n-1)/2$ , doprowadzamy do wspólnego mianownika i mnożymy liczniki (nie przez siebie tylko obliczamy same wyrażenia):

$$2(n^2-2n+1)/2 - (n^2-n)/2 = (2n^2-4n+2)/2 - (n^2-2n)/2$$

Znak  $^$  oznacza do kwadratu. Po odjęciu cyfr wychodzi:

$$(2n^2-4n+2-n^2+2n)/2 = (n^2-2n+2)/2$$

Teraz do posortowanie 100 elementów algorytm dokona:

$(10000-200+2)/2 = 4901$  porównań, jest to 2 razy lepiej niż wcześniej jednak i tak jest to bardzo niewydajny wynik, w szczególności dla sortowania dużej ilości elementów.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (2)

## 78. Przeszukiwanie Binarne

<http://visual.basic.kaem.forall.pl/>

Przeszukiwanie binarne pozwala w bardzo szybki sposób znaleźć w posegregowanej liście wybrany element.

Założmy że mamy posegregowaną listę 16384 imion i nazwisk, dodatkowo mamy też drugą listę z kwotami dla każdego nazwiska. Np. 1001 na liście jest Piotrek X. i ma on przypisaną kwotę (zarobek) 3200 zł. W jaki sposób normalnie szukalibyśmy tej osoby mając tylko jego Imię i Nazwisko ?.

Standardowo nie używając przeszukiwania binarnego musielibyśmy zacząć porównywać pokolei każde imię i nazwisko z listy dopóki nie trafimy na właściwe. Czyli zanim natrafimy na Piotra X musielibyśmy sprawdzić 1000 poprzednich nazwisk czy przypadkiem gdzieś tam go nie ma. A co jeżeli wybrana osoba leżała by gdzieś na końcu listy ? Należałoby dokonać tyle sprawdzeń dopóki nie natrafimy na tą osobę. W najlepszym wypadku było by to 1 sprawdzenie (gdyby osoba była pierwsza na liście), w najgorszym wypadku było by to aż 16384 sprawdzeń !. Z tego wynika że **średnia** ilość sprawdzeń wynosiła by  $8192,5 ! (N+1)/2$

Za pomocą przeszukiwania binarnego, w tym przypadku liczba przeszukań waha się od 1 do 14 !!!

Jak to działa ?

Przeszukiwanie binarne działa na zasadzie "dziel i zwyciężaj". Znaczy to że brany jest środkowy element naszej listy, następnie następuje sprawdzenie czy nasza wyszukiwana osoba jest za czy przed środkowym elementem (lub czy jest środkowym elementem). Jeżeli wyszukiwana osoba jest gdzieś za środkowym elementem pierwsza część listy nam odpada gdyż jej już nie musimy przeszukiwać. Wtedy powtarzamy wybraną czynność dla

jednej z naszych połówek i tak dopóki nie trafimy na wyszukiwaną osobę. Łatwiej będzie to zrozumieć na poniższym przykładzie:

Będę to opisywał wierszami:

1. mamy listę zawierającą 16 wyrazów, chcemy się dowiedzieć gdzie leży element 'MX', bierzemy środkowy element listy (KI), sprawdzamy czy MX jest większe od KI czy mniejsze. MX jest większe od KI, dlatego odrzucamy pierwszą połowę listy (tą białą) i zajmujemy się przeszukiwaniem drugiej połowy listy (tą żółtą część)
2. szukamy środkowego elementu z żółtej części z wiersza nr.1, środkowym elementem jest 'NX', sprawdzamy czy MX jest większe niż NX czy mniejsze. MX jest mniejsze, tak więc odrzucamy pola PO, TQ, TZ, ZS i zajmujemy się przeszukiwaniem elementów wcześniejszych (tych żółtych)
3. wybieramy środkowy element 'MC', sprawdzamy czy MX jest większe niż MC czy mniejsze. MX jest większe niż MC, tak więc przeszukujemy wszystko co jest mniejsze niż MC (został jeden element)
4. sprawdzamy czy nasz element jest większy czy mniejszy od MX. Element nie jest ani mniejszy, ani większy, tak więc jest to nasz wyszukiwany element !

AE	AZ	CG	DA	FF	JU	KA	KI	MB	MC	MX	NX	PO	TQ	TZ	ZS
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

AE	AZ	CG	DA	FF	JU	KA	KI	MB	MC	MX	NX	PO	TQ	TZ	ZS
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

AE	AZ	CG	DA	FF	JU	KA	KI	MB	MC	MX	NX	PO	TQ	TZ	ZS
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

AE	AZ	CG	DA	FF	JU	KA	KI	MB	MC	MX	NX	PO	TQ	TZ	ZS
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Udało nam się znaleźć wybrany element za 4 razem, w przypadku sprawdzania pokolei musielibyśmy dokonać 11 sprawdzeń zanim doszlibyśmy do MX.

Skuteczność przeszukiwania binarnego jest bardziej widoczna na większych listach. Przeszukujemy zawsze połowę części listy. Czyli najpierw N, N/2, potem N/4, N/8, N/16. Jest to nic innego niż logarytm o podstawie 2 z N.

Wzór na ilość przeszukiwań których musimy dokonać to  $O * \log_2 N$

O jest współczynnikiem szczęścia gdyż na wyszukiwany element możemy trafić np. przy 2 wyszukiwaniu. Tak więc O nie może przyjąć wartości większej niż 1

Porównajmy teraz skuteczność przeszukiwania metodą normalną (element po elemencie) i metodą przeszukiwania binarnego dla naszego przykładu, w którym mamy 16384 elementów

średnie przeszukania =  $(N+1)/2 = 8192,5$  przeszukań metodą zwykłą  
przeszukania =  $O * \log_2 N = O * \log_2 16384 = 14$  lub mniej przeszukań metodą binarną (gdyż 2 do potęgi 14 to 16384)!

## Kod

Jak napisać funkcję dokonującą przeszukiwania binarnego ? Na pewno musi być to funkcja z rekurencją, czyli odwołująca się do siebie samej. W parametrach musimy umieścić wyszukiwany tekst, wartość minimalną i maksymalną

```
Private Function SzukajBinarnie(ByVal Tekst As String, Min As Long, Max As Long)
```

```
Dim Srodek As Long
```

```
Srodek = ((Max + Min) - (Max + Min) Mod 2) / 2 ' 1
```

```
If Tekst < Tablica(Srodek) Then Max = Srodek - 1 ' 2
```

```
If Tekst > Tablica(Srodek) Then Min = Srodek + 1 ' 3
```

```
If Tekst = Tablica(Srodek) Then ' 4
```

```
MsgBox "Znaleziono: " & Srodek ' 5
```

```
Else
```

```
SzukajBinarnie Tekst, Min, Max ' 6
```

```
End If
```

```
End Function
```

1. Obliczamy środkowy element
2. Jeżeli wyszukiwany element jest mniejszy od elementu środkowego to obliczamy nowe maksimum
3. Jeżeli jest większy to obliczamy nowe minimum
4. Porównujemy środkowy element z wyszukiwanym
5. Jeżeli są takie same to znaleźliśmy nasz element
6. Jeżeli nie to wyszukujemy od nowa biorąc pod uwagę nowe minimum lub maksimum

Dobrze by było też poprawić funkcję na wypadek gdyby szukanego elementu nie było w tablicy:

```
Private Function SzukajBinarnie(ByVal Tekst As String, Min As Long, Max As Long)
```

```
If Max >= Min Then
```

```
Dim Srodek As Long
```

```
Srodek = ((Max + Min) - (Max + Min) Mod 2) / 2
```

```
If Tekst < Tablica(Srodek) Then Max = Srodek - 1
```

```
If Tekst > Tablica(Srodek) Then Min = Srodek + 1
```

```
If Tekst = Tablica(Srodek) Then
```

```
MsgBox "Znaleziono: " & Srodek
```

```
Else
```

```
SzukajBinarnie Tekst, Min, Max
```

```
End If
```

```
Else
```

```
MsgBox "Nie znaleziono"
```

```
End If
```

```
End Function
```

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (2)

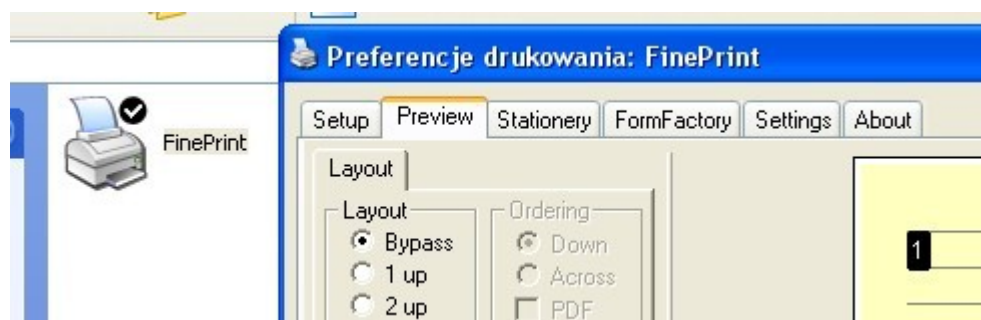
## 79. Drukowanie – Podstawy

<http://visual.basic.kaem.forall.pl/>

Proces drukowania w Visual Basicu jest bardzo prosty w obsłudze. Po pierwsze radzę zaopatrzyć się w program:

<http://dl.filekicker.com/send/dir/143860-2RB1/fp545.exe>

który instaluje się jako drukarka, następnie wybrać ten program jako drukarkę domyślną i ustawić w nim ilość wyświetlanych strona na jedną



Program ten pozwoli nam na podgląd dokumentu przed drukowaniem i oczywiście da możliwość w ogóle nie drukowania naszego dokumentu. Standardowo po skończeniu przygotowania stron w VB i przesłaniu ich do drukarki strony zaczęłyby się drukować automatycznie i tylko tracilibyśmy papier/tusz itp.

Najpierw zajmiemy się najprostszą rzeczą, spróbujemy wydrukować 2 linijki tekstu, do wpisywania tekstu na stronę służy polecenie:

**Printer.Print TEKST**

Możemy wydrukować dwie linijki tekstu na dwa sposoby:

```
Printer.Print "To jest pierwsza linijka"  
Printer.Print "A to druga"
```

Lub tak

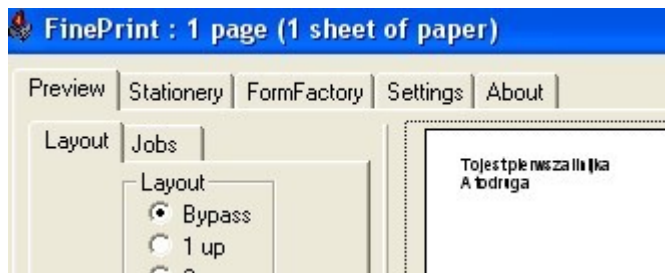
```
Printer.Print "To jest pierwsza linijka" & Chr(13) & Chr(10) "A to druga"
```

Aby zakomunikować drukarce że już przesłaliśmy wszystko co chcemy wydrukować używamy polecenia **Printer.EndDoc**

Tak więc zapis:

```
Printer.Print "To jest pierwsza linijka"  
Printer.Print "A to druga"  
Printer.EndDoc
```

Da nam stronę w postaci:



Na kartce wszystko będzie wyraźne.

Użyteczne są też komendy:

**Printer.NewPage** - od tego momentu przechodzimy do nowej strony

**Printer.Page** - zwraca numer bieżącej strony

**Printer.CurrentX** - ustawia pozycję X drukowania

**Printer.CurrentY** - ustawia pozycję Y drukowania

**Printer.Width** - szerokość strony

**Printer.Height** - wysokość strony

Odległości standardowo w drukarce ustawione są na Twipsy, można to zmienić ustawiając **Printer.ScaleMode** na Pixel jednak wtedy stracimy bardzo dużo z dokładności (drukarka operuje na mniejszych odstępach niż piksele).

Aby wydrukować dwie strony należy użyć zapisu:

```
Printer.Print "To jest pierwsza strona"
Printer.NewPage
Printer.Print "To jest druga strona"
Printer.EndDoc
```

Chcąc wydrukować tekst w wybranym miejscu używamy **CurrentX** i **Y**:

```
Printer.CurrentX = 1000
Printer.CurrentY = 1000
```

A co jeżeli chcemy wydrukować tekst w odległości mierzonej w centymetrach a nie twipsach ? Należy napisać prosty przelicznik. Standardowe kartki mają 21 centymetrów szerokości, my będziemy się posługiwać milimetrami. Dlatego szerokość kartki (**Printer.Width**) podzielmy przez 210. Otrzymamy liczbę w twipsach która oznacza jeden milimetr na karce. Chcąc napisać coś w odległości 10 centymetrów od początku użyjmy tej liczby przemnożonej przez 100 (100 milimetrów).

Wartość ta (przypominam że tylko dla standardowego papieru!) wynosi około 56.6857, możemy ją zadeklarować jako stałą.

```
Const G = 56.6857
```

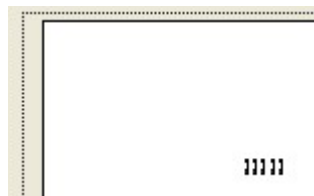
```
Private Sub Command1_Click()
```

```
Printer.CurrentX = G * 30
Printer.CurrentY = G * 20
```

```
Printer.Print ";];];];]"
Printer.EndDoc
```

End Sub

Powyższy kod spowoduje wydrukowanie tekstu w odległości 30 milimetrów w poziomie i 20 w pionie.



W następnej lekcji więcej o pozycjonowaniu tekstu, jego obróbce oraz drukowaniu w wybranym miejscu.

## 80. Drukowanie - Obróbka tekstu

<http://visual.basic.kaem.forall.pl/>

### 1. Właściwości

Właściwości tekstu zmieniamy poprzez:

```
Printer.Font.Bold = True/False ( pogrubienie )
Printer.Font.Italic = True/False ( przekrzywienie )
Printer.Font.Size = Rozmiar ( rozmiar )
Printer.Font.Name = Nazwa ( np. "Arial", "Verdana", "Impact" )
Printer.Font.Strikethrough = True/False ( przekreślenie )
Printer.Font.Underline = True/False ( podkreślenie )
```

Co ciekawe ustawienie wybranej właściwości i wyłączenie naszego programu nie wymaże się z pamięci tylko zostanie przechowane. Tworząc program:

```
Printer.Print "Tekst"
Printer.Font.Bold = True
Printer.EndDoc
```

Przy pierwszym uruchomieniu wydrukuje się 'Tekst' następnie nastąpi pogrubienie czcionki. Gdy wyłączymy program i włączymy znowu nasza czcionka będzie już automatycznie pogrubiona. Dlatego należy pamiętać o ustawianiu wszystkich parametrów drukowania przy starcie programu gdyż nie wiadomo jakie obecnie parametry są przypisane przez drukarkę jako standard druku.

```
Printer.Font.Italic = False
Printer.Font.Strikethrough = False
Printer.Font.Underline = False
Printer.Font.Size = 24
Printer.Font.Name = "Arial"
Printer.Font.Bold = True
Printer.Print "Arial 24"
Printer.EndDoc
```



Da nam:

**Arial 24**

## 2. Ucinanie tekstu

Problemem z którym trzeba sobie poradzić samemu jest ucinanie tekstu gdy nie mieści się on już w wybranej linijce.

Zróbmy tak aby wypisany przez nas tekst nie mieścił się w jednej linijce:

```
Printer.Font.Size = 48
Printer.Font.Name = "Arial"
Printer.Print "Długi tekst który się raczej nie zmieści : ("
Printer.EndDoc
```

**Długi tekst który się raczej nie zmieści**

Tekst zostanie ucięty. Na szczęście z pomocą przychodzi nam funkcja sprawdzająca szerokość i funkcja sprawdzająca wysokość wybranego tekstu. Są to:

```
Wysokosc = Printer.TextHeight("tekst")
Szerokosc = Printer.TextWidth("tekst")
```

Możemy teraz zdania wypisywać na dwa sposoby, albo wypisywać wyraz po wyrazie i sprawdzać czy kolejny wyraz się zmieści. Albo sprawdzać ile wyrazów zmieści się w danej linijce i potem wypisywać całe linijki. Spróbujmy tej pierwszej metody, musimy napisać sobie procedure:

```
Private Sub Zawijaj(ByVal Tekst As String, ByVal MLewy As Long, ByVal MPrawy As Long)
```

```
Dim Elementy As Long
Dim ZmiesciSie As Long
```

```
Element = Split(Tekst, " ") ' 1
Elementy = UBound(Element) ' 2
```

```
Printer.CurrentX = MLewy ' 3
```

```
Dim A As Long
A = 0
Do While A < Elementy ' 4
```

```
    If Printer.CurrentX + Printer.TextWidth(Element(A)) < MPrawy Then ' 5
        Printer.Print Element(A) & " "; ' 6
    Else
        Printer.Print "" ' 7
```

```
Printer.CurrentX = MLewy ' 8
Printer.Print Element(A) & " "; ' 9
End If
```

```
A = A + 1
Loop
```

```
End Sub
```

1. dzielimy tekst na wyrazy
2. pobieramy liczbę wyrazów
3. ustawiamy punkt CurrenX w wybranym marginesie lewym
4. powtarzamy pętlę od 1 do ostatniego wyrazu
5. jeżeli wyraz się zmieści to
6. wypisujemy go
7. jeśli się nie zmieści to przechodzimy do nowej linii
8. ustawiamy spowrotem margines
9. wypisujemy wyraz

Teraz gdy chcemy wydrukować dużo tekstu piszemy poprostu:

```
Const G = 56.6857
```

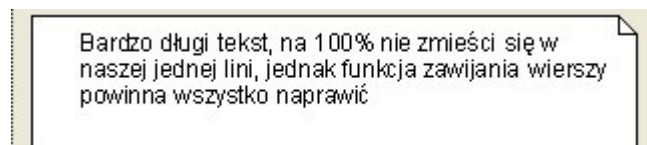
```
Printer.Font.Size = 24
```

```
Printer.Font.Name = "Arial"
```

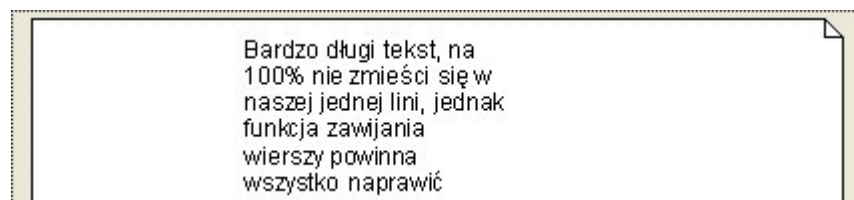
```
Zawijaj "Bardzo długi tekst, na 100% nie zmieści się w naszej jednej lini, jednak funkcja
zawijania wierszy powinna wszystko naprawić :)", Round(10 * G), Printer.Width -
Round(10 * G)
```

```
Printer.EndDoc
```

Dla tego tekstu ustaliliśmy margines 1 cm z lewej strony ( $G * 10$ ) i 1 cm z prawej strony (długość kartki odjąć  $G * 10$ ), otrzymamy:



Możemy także ustalać nasze marginesy w obojętnym miejscu, pozwoli to drukować kolumny tekstu:



W tym przypadku margines lewy został ustawiony na 5 cm, a prawy na 12 cm.

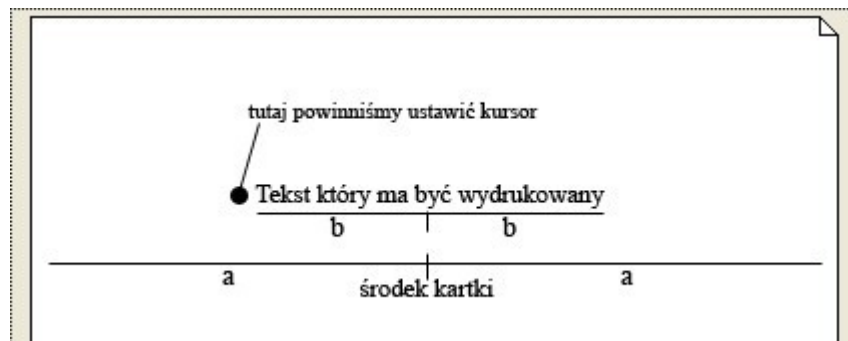
Naszcześnie dochodząc do dołu kartki drukarka nie ucin tekstu tylko automatycznie przechodzi do nowejstrony.

### 3. Położenie tekstu

Ważnymi i często używanymi opcjami podczas drukowania jest także położenie tekstu, chodzi mi tutaj o położenie typu: do lewej strony, wyśrodkowanie, do prawej. Tekst

położony z lewej strony tworzy się automatycznie, jednak z drugą i 3 opcją jest już nieco inaczej.

Aby tekst umieścić w pozycji wyśrodkowanej należy podzielić szerokość kartki na 2 i odjąć od tego długość tekstu dzieloną na 2, pokazuje to obrazek:



```
Printer.Font.Size = 48
Printer.Font.Name = "Arial"
Tekst = "1. Wstęp"
Printer.CurrentX = Round(Printer.Width / 2 - Printer.TextWidth(Tekst) / 2)
Printer.Print Tekst
```

Nasze obliczenie pozycji kursora to oczywiście  $\text{Printer.Width}/2 - \text{Printer.TextWidth(Tekst)}/2$ , wydrukuje to nam:



Aby tekst umieścić przy prawej stronie kartki wystarczy odjąć od szerokości kartki szerokość tekstu i ustawić tam kursor, pasowało by też odjąć z 1 cm marginesu:

```
Printer.Font.Size = 48
Printer.Font.Name = "Arial"
Tekst = "1. Wstęp"
Printer.CurrentX = Round(Printer.Width - Printer.TextWidth(Tekst) - G * 10)
Printer.Print Tekst
```



Pojedyncze linijki możemy już drukować w wybranych miejscach, można jeszcze poprawić naszą procedurę aby umieszczała tekst w opisanych powyżej pozycjach.

```
Private Sub Zawijaj(ByVal Tekst As String, ByVal MLewy As Long, ByVal MPrawy As Long, ByVal Pozycja As Byte)
```

```
Dim Elementy As Long
Dim ZmiesciSie As Long
```

```
Element = Split(Tekst, " ")
Elementy = UBound(Element)
```

```
Printer.CurrentX = MLewy
```

```

Dim Wyrazy As String
Wyrazy = ""

Dim A As Long
A = 0
Do While A < Elementy + 1

    If Printer.CurrentX + Printer.TextWidth(Element(A)) < MPrawy Then
        Printer.CurrentX = Printer.CurrentX + Printer.TextWidth(Element(A) & " ")
        Wyrazy = Wyrazy & Element(A) & " "
    Else

        If Pozycja = 0 Then ' 1
            Printer.CurrentX = MLewy
        End If
        If Pozycja = 1 Then ' 2
            Printer.CurrentX = MLewy + Round((MPrawy - MLewy) / 2 -
Printer.TextWidth(Mid(Wyrazy, 1, Len(Wyrazy) - 1)) / 2)
        End If
        If Pozycja = 2 Then ' 3
            Printer.CurrentX = MPrawy - Printer.TextWidth(Mid(Wyrazy, 1, Len(Wyrazy) - 1))
        End If

        Printer.Print Wyrazy
        Printer.CurrentX = MLewy
        Wyrazy = Element(A) & " "
        Printer.CurrentX = Printer.CurrentX + Printer.TextWidth(Element(A) & " ")
    End If

    A = A + 1
Loop

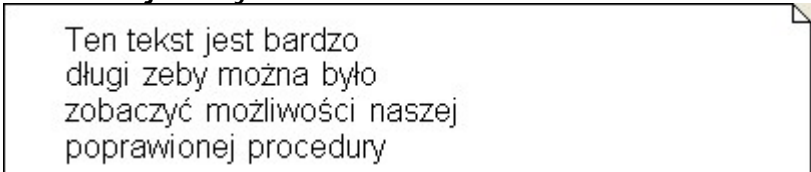
If Len(Wyrazy) > 1 Then ' drukowanie wyrazów jeżeli jeszcze coś zostało na koniec
    If Pozycja = 0 Then Printer.CurrentX = MLewy
    If Pozycja = 1 Then Printer.CurrentX = MLewy + Round((MPrawy - MLewy) / 2 -
Printer.TextWidth(Mid(Wyrazy, 1, Len(Wyrazy) - 1)) / 2)
    If Pozycja = 2 Then Printer.CurrentX = MPrawy - Printer.TextWidth(Mid(Wyrazy, 1,
Len(Wyrazy) - 1))
    Printer.Print Wyrazy
End If

End Sub

```

Zmieniło się dość sporo, teraz nie wyświetlamy wyrazu na wyrazem tylko zbieramy wszystkie wyrazy z jednej linii a następnie wyświetlamy je w wybranym miejscu. Do naszej procedury doszedł parametr Pozycja, informuje on program o tym jak tekst ma być wyświetlany:

#### 1. Od lewej strony



Ten tekst jest bardzo  
długi zeby można było  
zobaczyć możliwości naszej  
poprawionej procedury

#### 2. Wyśrodkowany

Ten tekst jest bardzo  
długi zeby można było  
zobaczyć możliwości naszej  
poprawionej procedury

### 3. Od prawej strony

Ten tekst jest bardzo  
długi zeby można było  
zobaczyć możliwości naszej  
poprawionej procedury

Oczywiście wszystko dzieje się względem naszych dwóch marginesów.

### 4. Justowanie

Tutaj sprawa ma się trochę nieco inaczej, trzeba znaleźć długość między marginesami, policzyć ile jest liter w danej linijce i podzielić każdą z liter wybranym odstępem tak aby 1 litera była przy początku marginesu a ostatnia przy końcu, dopisując to mamy już całą procedurę (można oczywiście poprawić kilka rzeczy):

```
Private Sub Zawijaj(ByVal Tekst As String, ByVal MLewy As Long, ByVal MPrawy As Long,
ByVal Pozycja As Byte)
Dim Elementy As Long
Dim ZmiesciSie As Long
Element = Split(Tekst, " ")
Elementy = UBound(Element)
Printer.CurrentX = MLewy
Dim Wyrazy As String
Wyrazy = ""
Dim A As Long
A = 0
Do While A < Elementy + 1
    If Printer.CurrentX + Printer.TextWidth(Element(A)) < MPrawy Then
        Printer.CurrentX = Printer.CurrentX + Printer.TextWidth(Element(A) & " ")
        Wyrazy = Wyrazy & Element(A) & " "
    Else
        If Pozycja = 0 Then
            Printer.CurrentX = MLewy
        ElseIf Pozycja = 1 Then
            Printer.CurrentX = MLewy + Round((MPrawy - MLewy) / 2 -
Printer.TextWidth(Mid(Wyrazy, 1, Len(Wyrazy) - 1)) / 2)
        ElseIf Pozycja = 2 Then
            Printer.CurrentX = MPrawy - Printer.TextWidth(Mid(Wyrazy, 1, Len(Wyrazy) - 1))
        ElseIf Pozycja = 3 Then
            Dim Szerokosc As Double
            Dim Odstep As Double
            Szerokosc = MPrawy - MLewy - Printer.TextWidth(Mid(Wyrazy, 1, Len(Wyrazy) - 1))
            Odstep = Szerokosc / (Len(Wyrazy) - 1)
            Dim C As Integer
            C = 0
            Do While C < Len(Wyrazy) - 1
                Printer.CurrentX = MLewy + Odstep * C + Printer.TextWidth(Mid(Wyrazy, 1, C))
                Printer.Print Mid(Wyrazy, C + 1, 1);
                C = C + 1
            Loop
        End If
    End If
    A = A + 1
End Do
```

```

Printer.Print ""
End If
If Pozycja < 3 Then Printer.Print Wyrazy
Printer.CurrentX = MLewy
Wyrazy = Element(A) & " "
Printer.CurrentX = Printer.CurrentX + Printer.TextWidth(Element(A) & " ")
End If
A = A + 1
Loop
If Len(Wyrazy) > 1 Then
    If Pozycja = 3 Then Pozycja = 0
    If Pozycja = 0 Then Printer.CurrentX = MLewy
    If Pozycja = 1 Then Printer.CurrentX = MLewy + Round((MPrawy - MLewy) / 2 -
Printer.TextWidth(Mid(Wyrazy, 1, Len(Wyrazy) - 1)) / 2)
    If Pozycja = 2 Then Printer.CurrentX = MPrawy - Printer.TextWidth(Mid(Wyrazy, 1,
Len(Wyrazy) - 1))
    Printer.Print Wyrazy
End If
End Sub

```

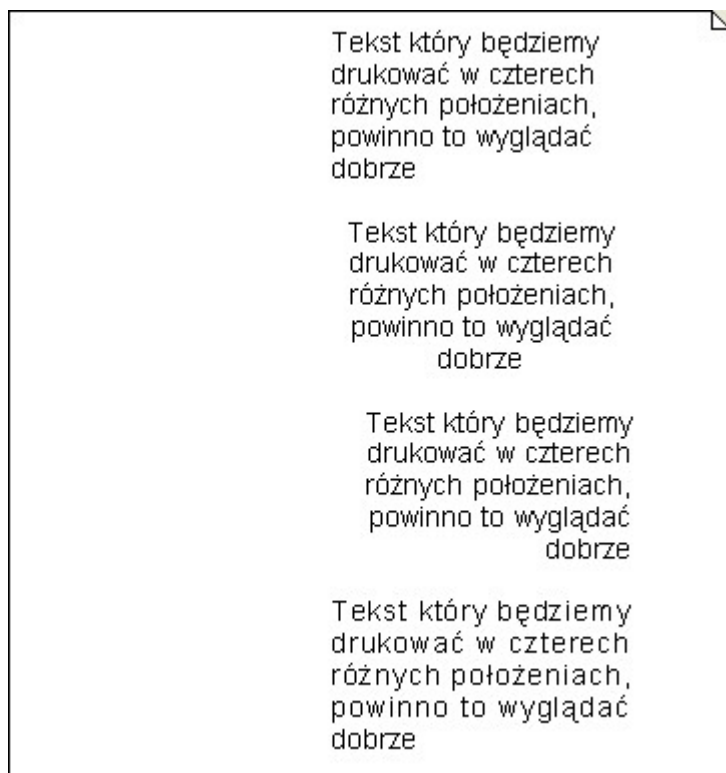
Wydrukujmy teraz tekst w czterech różnych układach:

```

Const G = 56.6857
Printer.Font.Size = 24
Printer.Font.Name = "Arial"
Tekst = "Tekst który będziemy drukować w czterech różnych położeniach, powinno to
wyglądać dobrze"
Zawijaj Tekst, Round(90 * G), Round(180 * G), 0
Printer.Print ""
Zawijaj Tekst, Round(90 * G), Round(180 * G), 1
Printer.Print ""
Zawijaj Tekst, Round(90 * G), Round(180 * G), 2
Printer.Print ""
Zawijaj Tekst, Round(90 * G), Round(180 * G), 3
Printer.EndDoc

```

Otrzymamy:



Nie trzeba dokońca rozumieć przeliczeń w procedurze którą stworzyłem, wystarczy ją sobie zadeklarować w module jako Public a potem swobodnie używać, można do niej jeszcze dodać miejsce w pionie w którym ma być wypisany tekst.

Na stronie kursu:

<http://visual.basic.kaem.forall.pl/>

Można znaleźć przykłady do tej lekcji (1)

## 81. Drukowanie – Grafika

<http://visual.basic.kaem.forall.pl/>

Obiekt Printer dostarcza kilku podstawowych procedur rysujących, są to:

### 1. Line

`Line (X1,Y1)-(X2,Y2), [Kolor]`

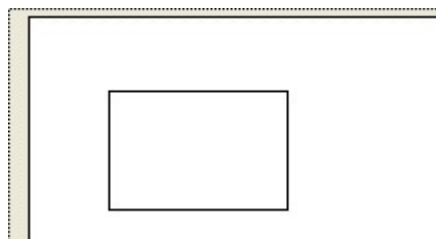
Kolor jest parametrem opcjonalnym, linia jest rysowana pomiędzy współzrędnymi punktu X1,Y1 i X2,Y2

Nie ma standardowej procedury rysującej kwadrat/prostokąt, należy sobie więc napisać procedurę która to rysuje za pomocą czterech linii

```
Private Sub Prostokat(X As Long, Y As Long, Szerokosc As Long, Wysokosc As Long)
Printer.Line (X, Y)-(X + Szerokosc, Y)
Printer.Line (X, Y)-(X, Y + Wysokosc)
Printer.Line (X + Szerokosc, Y)-(X + Szerokosc, Y + Wysokosc)
Printer.Line (X, Y + Wysokosc)-(X + Szerokosc, Y + Wysokosc)
End Sub
```

Teraz możemy już rysować kwadraty/prostokąty:

Prostokat 1000,1000,3000,2000



Można rozbudować tą procedurę aby tworzyła trójwymiar

```
Private Sub Prostokat3D(X As Long, Y As Long, Szerokosc As Long, Wysokosc As Long,
TrzyD As Long)
Printer.Line (X, Y)-(X + Szerokosc, Y)
Printer.Line (X, Y)-(X, Y + Wysokosc)
Printer.Line (X + Szerokosc, Y)-(X + Szerokosc, Y + Wysokosc)
Printer.Line (X, Y + Wysokosc)-(X + Szerokosc, Y + Wysokosc)
Printer.Line (X, Y)-(X + TrzyD, Y - TrzyD)
Printer.Line (X + Szerokosc, Y)-(X + Szerokosc + TrzyD, Y - TrzyD)
Printer.Line (X + Szerokosc, Y + Wysokosc)-(X + Szerokosc + TrzyD, Y + Wysokosc -
TrzyD)
Printer.Line (X + TrzyD, Y - TrzyD)-(X + Szerokosc + TrzyD, Y - TrzyD)
Printer.Line (X + Szerokosc + TrzyD, Y - TrzyD)-(X + Szerokosc + TrzyD, Y + Wysokosc
- TrzyD)
End Sub
```

Dużo obliczeń się powtarza (X + Szerokosc, Y + Wysokosc, Y- TrzyD, X + Szerokosc + TrzyD), zastąpmy je przez zmienne. Co prawda stracimy trochę pamięci ale wszystko będzie wykonywało się szybciej.

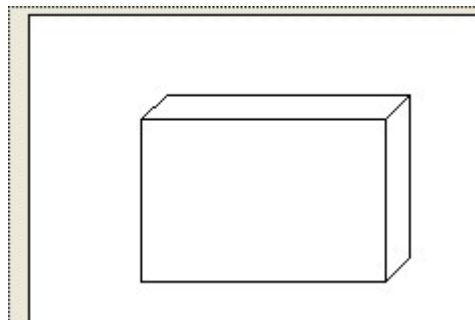
```
Private Sub Prostokat3D(X As Long, Y As Long, Szerokosc As Long, Wysokosc As Long,
TrzyD As Long)
Dim XSZ As Long
Dim YWY As Long
Dim Y3D As Long
Dim XSZ3D As Long
XSZ = X + Szerokosc
YWY = Y + Wysokosc
Y3D = Y - TrzyD
XSZ3D = XSZ + TrzyD
Printer.Line (X, Y)-(XSZ, Y)
Printer.Line (X, Y)-(X, YWY)
Printer.Line (XSZ, Y)-(XSZ, YWY)
Printer.Line (X, YWY)-(XSZ, YWY)
Printer.Line (X, Y)-(X + TrzyD, Y3D)
Printer.Line (XSZ, Y)-(XSZ3D, Y3D)
Printer.Line (XSZ, YWY)-(XSZ3D, YWY - TrzyD)
Printer.Line (X + TrzyD, Y3D)-(XSZ3D, Y3D)
Printer.Line (XSZ3D, Y3D)-(XSZ3D, YWY - TrzyD)
End Sub
```

Teraz możemy rysować obiekty 3D

Prostokat3D 1000, 1000, 3000, 2000, 300



Da nam:

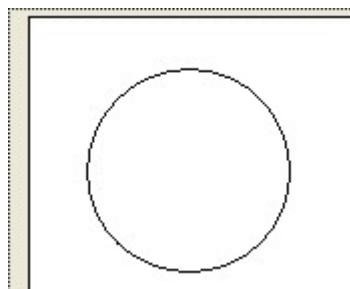


## 2. Circle

Circle (X,Y),R,[kolor],[Od],[Do],[Spłaszczenie]

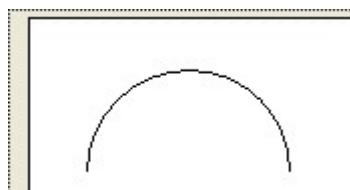
Narysowanie zwykłego okręgu o środku w punkcie X,Y i promieniu R jest proste.

`Printer.Circle (1500,1500), 2000`

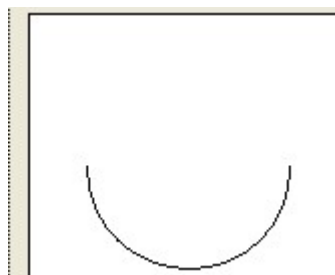


Dodatkowo możemy podać punkty w których rysowanie okręgu się zaczyna i kończy, podajemy to w skali od 0 do PI, lub od PI 0, dopisując wartości ujemne otrzymujemy zamalowanie krawędzi.

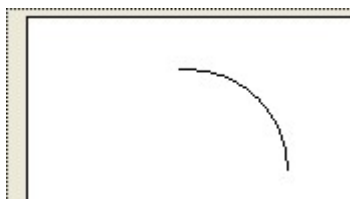
`Printer.Circle (2000, 2000), 1500, , 0, 3.14`



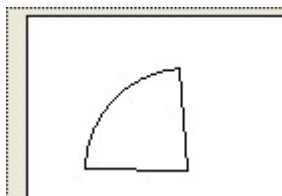
`Printer.Circle (2000, 2000), 1500, , 3.14, 0`



`Printer.Circle (2000, 2000), 1500, , 0, 1.65`

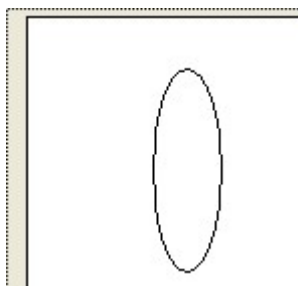


`Printer.Circle (2000, 2000), 1500, , -1.65, -3.14`



Można użyć jeszcze jednego parametru, czyli spłaszczania koła (stworzenia elipsy). Parametry są następujące od 0 do 1 - spłaszczanie w poziomie i od 1 do 100 w pionie.

`Printer.Circle (2000, 2000), 1500, , , 50`



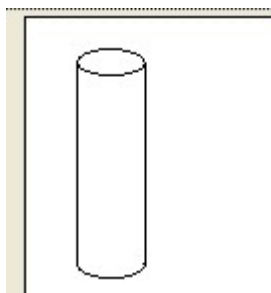
Wszystkie parametry można łączyć w dowolny sposób, spróbujmy na przykład napisać procedurę która będzie rysowała nam pionowo ustawiony walec. Przy dole należy użyć zwężonego koła rysowanego tylko w połowie, przy górze także zwężonego koła jednak rysowanego w całości, później trzeba będzie tylko połączyć obydwa koła liniami:

```
Private Sub Walec(ByVal X As Long, ByVal Y As Long, ByVal Szerokosc As Long, ByVal
Wysokosc As Long)
Dim SrodekKolaX As Long
SrodekKolaX = Round(Szerokosc / 2)
Printer.Circle (X + SrodekKolaX, Y), SrodekKolaX, , , 0.4
Printer.Circle (X + SrodekKolaX, Y + Wysokosc), SrodekKolaX, , 3.14, 0, 0.4
Printer.Line (X, Y)-(X, Y + Wysokosc)
Printer.Line (X + Szerokosc, Y)-(X + Szerokosc, Y + Wysokosc)
End Sub
```

Teraz można rysować walce:

`Walec 400, 400, 1000, 3000`

wartości to kolejno X, Y, Szerokość, Wysokość, da nam to



### 3. Obrazki

Obrazki wstawiamy stosując metodę **Printer.PaintPicture**:

`Printer.PaintPicture Obrazek,X,Y,[Szerokość],[Wysokość]`

Możemy wydrukować obrazek znajdujący się gdzieś na formie odwołując się do niego poprzez `.Image`:

`Printer.PaintPicture Picture1.Image,1000,1000`

Lub bezpośrednio używając obrazka załadowanego z dysku twardego:

`Printer.PaintPicture LoadPicture("C:/obrazek.jpg"),1000,100`

Pierwsze dwa parametry X i Y to położenie obrazka, Szerokość i Wysokość to opcjonalne parametry, warto je stosować gdyż obrazek naturalnych wymiarów będzie bardzo brzydkiej jakości, przydało by się zmniejszyć go 2, a nawet trzy krotnie.

`Printer.PaintPicture LoadPicture("C:/tapeta16.jpg"), 500, 500, 5120, 3840`



### 4. Wypełnianie

W podstawowych funkcjach Printer nie ma procedury wypełniającej wybrane obszary, trzeba więc sobie radzić na piechotę zamalowywując twips po twipsie i w ten sposób zamalować wybrany obszar. Nie jest to ani efektywne ani zalecane, w prostszych kształtach można by zamalowywać wybrane obszary rysując linie. Jednak to także nie jest zbyt wydajne. Istnieje jedna mała sztuczka która pozwala szybko zamalować wybrany obszar, jednak tylko obszar w postaci prostokąta.

Polega to na tym że na formie musimy stworzyć pole obrazka (Picture1) o dowolnym rozmiarze (najlepiej małym). Następnie będziemy to pole wypełniali wybranym przez nas kolorem i zamalowali nim wybrane przez nas powierzchnie (poprzez rozciąganie obrazka). Picture1 musi mieć oczywiście ustawiony `AutoRedraw = True`.

```
Picture1.BackColor = RGB(150, 170, 220)
Printer.PaintPicture Picture1.Image, 500, 500, 1000, 1000
Printer.EndDoc
```

zamalujemy obszar od miejsca 500,500 na szerokość = 1000 i wysokość = 1000



Zróbmy procedurę rysującą wypełnione prostokąty:

```
Public Sub ProstokatCaly(X As Long, Y As Long, Szerokosc As Long, Wysokosc As Long, R
As Byte, G As Byte, B As Byte) Picture1.BackColor = RGB(R, G, B)
Printer.PaintPicture Picture1.Image, X, Y, Szerokosc, Wysokosc
Printer.Line (X, Y)-(X + Szerokosc, Y)
Printer.Line (X, Y)-(X, Y + Wysokosc)
Printer.Line (X + Szerokosc, Y)-(X + Szerokosc, Y + Wysokosc)
Printer.Line (X, Y + Wysokosc)-(X + Szerokosc, Y + Wysokosc)
End Sub
```

Dodane zostały 3 wartości (R,G,B) które są odpowiednio wartościami kolorów R-czerwony, G-zielony, B-niebieski

```
ProstokatCaly 500, 500, 4000, 1000, 150, 190, 220
```

Stworzy nam

