1 SPIS TREŚCI

2 Tworzenie nowego programu

Zakładam, że programy będą tworzone w środowisku Microsoft Office 2003. Najlepiej korzystać z programu Excel ponieważ takie podejście pozwala wykonać obliczenia pomocnicze lub sprawdzić poprawność działania naszego programu.

Wybieramy ***Narzędzia > Makro > Edytor Visual Basic*** (Alt+F11).

Do rozpoczęcia pracy (pisania procedury) konieczne jest wstawienie obiektu zwanego UserForm (Okno formularza), na który będziemy mogli wstawiać kontrolki i definiować ich właściwości i działania. ***Insert > UserForm***.

Na **UserForm** wstawiamy kontrolki z paska narzędzi **ToolBox**. Pierwszą kontrolką, której będziemy używać jest ***CommandButton***.

3 Właściwości przycisku CommandButton

(name)	W tym polu wpisujemy nazwę identyfikującą nasz przycisk – najlepiej używać skrótu cmd – tak wię zmieniamy nazwę naszego przycisku na cmdwitaj
Caption	Chyba najważniejsza – Tu ustalamy wyrazy które mają znaleźć się na naszym przycisku – wpisujemy Witaj !
Enabled	Tu ustalamy czy przycisk będzie się dało wcisnąć czy też nie – my oczywiście zostawiamy True (Prawda)
Font	W tym oknie możemy ustalić czcionkę, wielkość i efekty dla napisu
Height	Wysokość przycisku. Raczej wygodniejsze jest rozciąganie przycisku bezpośrednio na formularzu
Index	Ustala kolejność przeskakiwania focusa. Chodzi o to gdy np. na formularzu ustawimy trzy przyciski o nazwie cmdJeden, cmdDwa, cmdTrzy i kolejno ustawimy na nie index 1,3,2 to przeskakując „Tab-em” kolejno podświetlą się przycisk cmdJeden (1), cmdTrzy (2), cmdDwa (3). Na razie zostawiamy puste.
Left	Określa odległość przycisku od lewej krawędzi ekranu. Wygodniej jest przemieszczać przycisk bezpośrednio na formie.
Style	Oznacza czy w przycisku będziemy dokonywać przeróbek kosmetycznych – dla eksperymenu zaznaczmy True
BackColor	Teraz wracamy się prawie do samego początku. Możemy tu zmienić kolor przycisku. Zmieńmy go np. na kolor czerwony (znajduje się w zakładce pallete)
Top	Określa położenie przycisku od górnej krawędzi ekranu
Width	Określa szerokość przycisku
Visible	Określa czy przycisk ma być widoczny na formularzu. Jeśli zaznaczymy false, to po uruchomieniu programu zobaczymy pusty formularz. Zostaw więc true

4 Składnia funkcji odpowiedzialnej za wyświetlenie okienka dialogowego

```
X = MsgBox(„Jakiś tam tekst”,ikonka + przyciski, „Tytuł”)
```

Ikonki

vbCritical	Ikonka krytyczna (przekreślone czerwone kółeczko z charakterystycznym dźwiękiem)
vbQuestion	Ikonka pytania (Dymek, a w nim pytajnik)
vbExclamation	Ikonka ostrzegawcza (Żółty trójkąt z wykrzyknikiem)
vbInformation	Ikonka informacji (Dymek a w nim literka „i”)
vbSystemModal	Brak ikonki. Za to na pasku tytułowym pojawia się ikonka windows

Przyciski

vbOKonly	Tylko OK. – nie trzeba tego pisać. On i tak jest używany domyślnie
vbOKCancel	Przycisk OK. i Anuluj
vbAbortRetryIgnore	Przyciski Przerwij, Ponów próbę i Zignoruj
vbYesNoCancel	Przyciski Tak, Nie, Anuluj
vbYesNo	Tylko przyciski Tak i Nie
vbRetryCancel	Tylko przyciski Ponów i Anuluj

MsgBox zwraca:

1	vbOK
2	vbCancel
3	vbAbort
4	vbRetry
5	vbIgnore
6	vbYes
7	vbNo

Ćwiczenie 1 do punktu 1 i 2

Napisz program z dwoma przyciskami Jeden o nazwie „Witaj” i kolorze żółtym, a drugi „Pytanko” i kolorze zielonym. Klikając na pierwszym powinien wyświetlić się komunikat „Jestem spoko” z ikonką informacyjną i przyciskiem OK. Zmień tytuł na „Przywitanie”. A drugi z tekstem „Czy mnie lubisz?” z ikonką pytania i przyciskami Tak – Nie, oraz o tytule „Małe pytanko”.

5 Operatory

Czyli znaki pozwalające wykonać jakieś operacje (matematyczne, logiczne, na tekście).

Podstawowe operatory arytmetyczne

Operator	Znaczenie	Przykład
+	Dodawanie	5+3
-	Odejmowanie	4-2
*	Mnożenie	3*7
/	Dzielenie	15/5
^	potęgowanie	3^2

Operatory do pracy z tekstem i łączące

Operator	Znaczenie	Przykład
&	(And), łączenie ciągów tekstowych	"zmienna x =" & x
+	łączenie ciągów tekstowych	Jeśli korzystamy ze zmiennych tekstowych to umożliwia łączenie tekstu Np. 5+2=52

Operatory logiczne - patrz rozdział *11 Operatory logiczne And , Or, Not.*

Operator przypisania

Operator	Znaczenie	Przykład
=	Przypisz np. x=5 co oznacza, że od tego momentu zmienna x ma wartość 5 Niestety w VB taki sam znak jest używany do przypisania jak i do sprawdzenia relacji i z kontekstu trzeba się domyślić, o który chodzi.	x=8

Operatory relacji

Operator	Znaczenie	Przykład
=	Porównanie dwóch liczb lub wartości zmiennych	X=4, czy x jest równe 5
<>	Czy liczby (zmienne) są różne	X<>5 , czy x jest różne od 5
<	Czy liczba jest mniejsza	X<5 , czy x jest mniejsze od 5
>	Czy liczba jest większa	X<5 , czy x jest mniejsze od 5
>=	Czy liczba jest większa bądź równa	X>=4
<=	Czy liczba jest mniejsza bądź równa	X<=5

6 Funkcje Dim i If

Funkcja Dim musi być wstawiona na początku procedury, żeby program pamiętał zmienną x

Funkcja IF sprawdza jaka jest wartość zmiennej x i w zależności od jej wartości wykonał odpowiednią operację.

Przykład:

```
Private Sub cmdPytanko_Click
Dim x
x = msgbox("Czy mnie lubisz ?",vbQuestion+vbYesNo,"Małe Pytanko")
If x=6 then
y = msgbox("Fajnie, że mnie lubisz ! ",vbExclamation,"Małe pytanko")
End If
If x=7 then
z = msgbox("Strasznie mi przykro...",vbExclamation,"Małe pytanko")
End IF
End Sub
```

Jak widać powyżej kliknięcie TAK przypisuje zmiennej X wartość = 6, a przy kliknięciu NIE x=7.

Funkcja **Dim** może przechowywać następujące wartości

Boolean	Przetrzymuje True lub False
Byte	Liczby całkowite z zakresu od 0 do 255
Currency	Liczby z zakresu -922337685477,5808 do + (-||-)
Date	Data od 01.01.0100 - 31.12.9999
Decimal	Liczba zapisana z dokładnością do 28 liczb po przecinku
Double	Liczba z zakresu -1,79769313486232E+308 do + (-||-)
Integer	Liczba z zakresu od -32768 do 32767
Single	Liczba z zakresu -3402823E+38 do + (-||-)
String	Dane znakowe (alfanumeryczne)
Wariant	Dowolne - nie musimy tego pisać gdyż Variant jest domyślnie

Funkcję DIM deklarujemy następująco (składnia funkcji Dim):

```
DIM x as typ
```

Poprzednio stosowaliśmy tylko Dim x (to wystarczy ponieważ program przyjmował domyślnie Variant).

Przy pisaniu większych programów ważne jest aby definiować jak najmniejszy zakres do zapamiętania np. tylko informację TRUE/FALSE (do zapisania tego program potrzebuje tylko jeden bit) albo Byte (do zapisania tego program potrzebuje tylko jednego bajta).

Ćwiczenie z funkcji Dim. Prześledź działanie procedury:

```
Dim x As Boolean
x = MsgBox("Co wybierasz ?", vbYesNo)
c = MsgBox(x, vbCritical)
```

Sprawdź jak ta procedura działa jak zamienisz parametr “Boolean” na „Byte”.

Ćw. 2 do punktu 3

Rozszerz możliwości procedury napisanej w ćwiczeniu 1. Po kliknięciu przycisku Witaj niech pojawi się pytanie „Czy jesteś zdrowy?” i z przyciskami TAK i NIE. Kliknięcie TAK niech wywoła komunikat „To fajnie” A kliknięcie NIE komunikat „To idź się leczyć!!!”

Drugi przycisk z tekstem „Czy mnie lubisz?” i przyciskami zmodyfikuj tak aby procedura reagowała na przyciski TAK i NIE wg Twojego uznania. Oczywiście cieszymy się z odpowiedzi TAK i martwimy odpowiedzią NIE.

Ćw. 3 do punktu 3

Napisz procedurę, która będzie się pytała użytkownika kolejno o następujące rzeczy:

1) Czy jesteś dorosły?
2) Czy jesteś kobietą?
3) Czy jesteś blondynką?
4) Czy masz samochód?

Po zadaniu wszystkich pytań wyświetl komunikaty, w którym wypiszesz kolejno wszystkie rzeczy których się dowiedziałeś/łaś od użytkownika.

Przykład:

Użytkownik odpowiedział na pytania kolejno: TAK, NIE, TAK, NIE.

Twój program powinien wyświetlić komunikaty:

Jesteś dorosły

Jesteś mężczyzną

Jesteś blondynką ;-)

Nie masz samochodu.

7 TextBox oraz Label (Etykieta)

Oba pola są polami tekstowymi przy czym w textbox można wpisywać coś z klawiatury a label jedynie wyświetla zadaną wartość, której użytkownik nie może modyfikować.

Wykonaj program z przykładu poniżej. Program ma obliczać kwadrat z podanej liczby.

Na formie (UserForm) ułóż **przycisk** - nadaj mu nazwę cmdLicz, a w Caption wpisz Licz

Liczbę będziemy wpisywać w pole tekstowe Jest to biały prostokąt z wpisanymi literkami "ab"

Wstaw pole **TextBox** i wyczyść je jeśli trzeba (pole text) zmień nazwę na txtLiczba

Wstaw pole **Label** jest widoczne na pasku narzędzi jako duża litera "A" napis Label1 - zmień na "Twój wynik to:" oczywiście w opcji Caption zmień nazwę na lblStalyTekst

Obok niego postaw drugie pole Label, nadaj mu nazwe lblWynik i wyczyść zawartość (Caption). Zmień etykietę Formy na "Do kwadratu".

Przyciskowi przypisz procedurę:

```
lblWynik.Caption = txtLiczba ^ 2
```

Jeśli chcielibyśmy dodawać liczby mogą być problemy - musisz programowi powiedzieć, że to co wpiszesz w pole tekstowe jest liczbą. Służy do tego funkcja CSng. Poniżej przykład jak jej użyć:

```
Private Sub cmddodaj_Click()
Dim a, b As Single
a = CSng(txta)
b = CSng(txtb)
lblwynik = a + b
End Sub
```

Przyjąłem, że pierwszy textbox nazywa sie txta a drugi txtb. W tym momencie przyporządkowujesz zmiennej "a" to co jest w textbox-ie1 jako liczbę. Pamiętaj, że w Textbox-ie jest zawsze tekst. Musisz ten tekst dopiero przerobić na liczbę, a dopiero potem wykonywać działania arytmetyczne

Ćwiczenie z zastosowania pól TextBox oraz Label

Napisz program z dwoma polami tekstowymi, czterema przyciskami i jednym polem label. Program ma wykonywać pewne działania matematyczne. Przyciski kolejno powinny wykonywać funkcje: dodawanie, odejmowanie, mnożenie i dzielenie liczb podanych w textbox-ach. Wynik oczywiście powinniśmy dostać w polu Label.

8 funkcja CSng

Ta funkcja jest odpowiedzialna za konwersję danych. Główne typy danych to String i Single. Wszelkie łańcuchy znaków Visual Basic interpretuje w cudzysłowach (""), czyli jako String, daty w ##, a liczby zostawia same.

Ćwiczenie

Wykonaj następujące procedury lblwynik = txtliczba1 + txtliczba2 , gdzie pola txtliczba1 i txtliczba2 są polami **TextBox** a pole lblwynik jest polem **Label.** Spróbuj wpisać w pola **TextBox** zarówno liczby jak i jakiś dowolny tekst

Następnie zamień procedurę na następującą.

```
lblwynik = CSng(txtliczba1) + CSng(txtliczba2)
```

Teraz też spróbuj wpisać w pola **TextBox** zarówno liczby jak i jakiś dowolny tekst

Procedura ta wykonuje konwersję danych tylko na potrzeby tego jednego obliczenia.

Bardziej elegancki sposób konwersji podałem poniżej. W tym sposobie procedura dodatkowo zapamiętuje zmienne a i b.

```
Dim a,b as single
a = CSng(txtliczba1)
b = CSng(txtliczba2)
lblwynik = a + b
```

Ćwiczenie: Wykonaj program wspomagający pracę handlowca w sklepie z wykładzinami podłogowymi. Program ma pobrać szerokość i długość sprzedawanej wykładziny i obliczyć pole powierzchni w m^2. Następnie pobrać cenę jednostkową wykładziny i obliczyć cenę zakupu całej wykładziny.

Rozwiązanie

9 InputBox.

Jest to komunikat, z wbudowanym polem tekstowym. Może nas np. zapytać o imię, a my mu od razu odpowiemy.

Utwórz program z przyciskiem. Wewnątrz niego umieść następujący kod:

```
Dim x as string

x = InputBox ("Jak masz na imię ?")
```

Trzeba jednak zmodyfikować funkcję, żeby zmienić tytuł. Składnia:

```
x = InputBox("Jak masz na imię?"," Tutaj tytuł", "Tu domyślna odpowiedź")
```

Teraz napisz program, który zapyta się Ciebie jak masz na imię, a następnie w MsgBox-ie Cię przywita "Witaj -imię-"

W przycisku umieszczamy następujący kod :

```
Dim x,y as string
x = InputBox ("Jak masz na imię ?","Przywitanie","Może Marek...")
y = MsgBox ("Witaj_" & x ,vbExclamation,"Przywitanie")
```

Znak & służy do łączenia łańcuchów (podobnie jak +). W tym momencie łączy on tekst "Witaj" z x-em czyli zmienną zawierającą nasze imię. Powstaje więc jeden ciąg znaków który widzimy w msgBox-ie.

ZADANIE

Napisz procedurę, która zada Ci wymyślone przez Ciebie pytanie a potem wykorzysta Twoją odpowiedź na to pytanie w jakimś komunikacie.

10 Funkcja IF

Napisz program, który da się uruchomić tylko po podaniu hasła.

W związku z tym, że o hasło musimy zapytać przed uruchomieniem programu nie możemy go przypisać do przycisku. Najlepiej procedurę pytającą o hasło przypisać do formularza (UserForm).

Zapytajmy więc o hasło:

```
Dim haslo as string
haslo = InputBox ("Proszę podać hasło dostępu","Hasło")
```

I sprawdźmy czy podane hasło jest poprawne

```
If haslo <> "SLASH" then
x = msgbox("Podałeś złe hasło",vbcritical,"Błąd")
End
Else
y = msgbox("Podałeś dobre hasło",vbExclamation,"OK")
End IF
```

Pamiętaj, że Polecenie END – to zamknięcie programu.

Else – rozszerzenie instrukcji IF – oznacza „w przeciwnym wypadku”

Przykład

Ćwiczenie: Wykonaj poniższą procedurę.

```
Dim wiek As Byte
Dim x, y, z As Byte
wiek = InputBox("Podaj, ile masz lat", "Pytanie o wiek", "17")
If wiek < 12 Then
x = MsgBox("Jesteś jeszcze za młody na pewne rzeczy", vbExclamation,
"Młody")
ElseIf wiek >= 40 Then
y = MsgBox("Jesteś w pełni wieku", vbExclamation, "Starszy")
Else
z = MsgBox("Jesteś jeszcze młody", vbExclamation, "Średni wiek")
End If
```

Ćwiczenie

Wykonaj program zabezpieczony przed uruchomieniem za pomocą hasła. Na userformie niech znajdzie się jeden przycisk i jedno pole tekstowe opisane za pomocą pola label. Do pola tekstowego użytkownik powinien wpisać swoja średnią ocen.

Program powinien działać w ten sposób, że po kliknięciu przycisku sprawdzi jaka średnia ocen jest wpisana w polu tekstowym i w zależności od jej wysokości wygeneruje odpowiedni komunikat.

Procedura powinna rozróżniać trzy różne stany:

od 1 do 2,5 "Bardzo niska średnia"

od 2,51 do 3,9 "Może być"

powyżej 3,91 "Bardzo ładna średnia"

Rozwiązanie

11 Operatory logiczne And , Or, Not.

And – i; Or – lub; Not - nie

Napisz procedurę, która sprawdzi dochody dwu firm. Niech pojawią się gratulacje, jeśli dochody jednej i drugiej firmy przekroczą 5000 zł.

Utwórz dwa pola tekstowe i przycisk. Pierwsze pole nazwij txtFirma1, a drugi txtFirma2, wyczyść ich zawartość (Text). Przycisk nazwij np. cmdWynik. Nadaj mu etykietę (Caption).

```
Dim x
IF txtfirma1 > 5000 and txtfirma2 > 5000 then
x = msgBox("Gratulacje",vbInformation,"Dochód")
End IF
```

Ćwiczenie: Teraz napisz procedurę, która pogratuluje nam w wypadku, jeśli którakolwiek z firm przyniesie dochód powyżej 5000zł.

Żeby zapoznać się z funkcją NOT wykonaj poniższą procedurę.

```
Dim x
IF NOT txtFirma1 < 5000 then
x = msgBox("Gratulacje",vbInformation,"Dochód")
End IF
```

Pamiętaj zawsze można napisać funkcję tak aby nie trzeba było używać instrukcji NOT.

12 Rozszerzenie funkcji InputBox

Dzięki poznaniu funkcji end jesteśmy w stanie spowodować aby procedura zareagowała na wciśnięcie Cancel. Wykonaj poniższy przykład:

```
Dim x,y as string
x = InputBox ("Jak masz na imię ?","Przywitanie","Może Marek...")
IF x = "" then end
y = MsgBox ("Witaj_" & x ,vbExclamation,"Przywitanie")
```

Jak można się domyślić kliknięcie Cancel nie przypisuje żadnej wartości zmiennej x. A co się stanie jeśli nie naciśniemy Cancel, ale nic nie wprowadzimy do pola **InputBox**?

Jeśli nie mamy zamiaru użyć instrukcji Else, lub ElseIF, a instrukcja warunkowa mieści się w jednej linijce nie trzeba rozpoczynać nowej linijki i co za tym idzie kończyć IF-a

Zadanie 1: Wykonaj program, który spyta się Czy masz samochód? (lub inny przedmiot). Możliwe odpowiedzi to

- tak

- nie

- anuluj (cancel)

Każda z tych odpowiedzi niech spowoduje inną reakcję programu np.

TAK – To fajnie że masz samochód,

NIE – Oj szkoda,

CANCEL – Czemu unikasz odpowiedzi na to pytanie?

Zadanie 2

Umieść na userformie przycisk i dwa pola tekstowe. W polach tekstowych będą wpisywane średnie ocen dwóch uczniów. Po kliknięciu przycisku (CommandButon) niech pojawi się Input Box z pytaniem o imię pierwszego ucznia - kliknięcie Cancel niech powoduje przypisanie uczniowi imienia **Bolek**, następnie niech pojawi się drugi InputBox z pytaniem o imię drugiego ucznia. Kliknięcie Cancel niech powoduje przypisanie temu uczniowi imienia **Lolek**.

Następnie program powinien sprawdzić zawartość obu pól tekstowych. Jeżeli obie średnie będą powyżej 4 niech pojawi się komunikat „Zarówno **Bolek** jak i **Lolek** mają super średnie ocen”. Jeśli średnia jednego z uczniów będzie niższa niż 4 to komunikat może brzmieć „Uczniowie **Bolek** i **Lolek** nie uzyskali średnich powyżej 4.”.

Oczywiście jeśli w polach inputbox zostaną wpisane jakieś imiona uczniów to powinny się one pojawić w komunikatach zamiast imion Bolek i Lolek.

Rozwiązanie

Sprawdzian z podstaw

13 Funkcja Select Case

Funkcję Select Case zawsze można zastąpić funkcją IF - jednak funkcja Case ma dużo krótszy zapis. Ponizej przykład programu pytającego nas o wiek. Uruchamiamy VB - tworzymy przycisk i dodajemy do niego kod:

```
Dim wiek As Currency
wiek = InputBox("Podaj swój wiek...", "Pytanie o wiek")
If wiek <= 0 Then
x = MsgBox("To jest niemożliwe", vbCritical, "Błąd")
ElseIf wiek < 11 Then
x = MsgBox("Mniej niż 11 - jeszcze dziecko", vbInformation, "Odp")
ElseIf wiek < 18 Then
x = MsgBox("Młodzież", vbInformation, "Odp")
ElseIf wiek < 40 Then
x = MsgBox("Dorosły  do 40", vbInformation, "Odp")
ElseIf wiek = 100 then
x = MsgBox("Piękny wiek !!!", vbInformation, "Odp")
ElseIf wiek < 110 Then
x = MsgBox("Pełnia wieku...", vbInformation, "Odp")
Else
x = MsgBox("Niemożliwe !!!", vbCritical, "Błąd")
End If
```

Na pierwszy rzut oka w tym kodzie są błędy - nie można kilku MsgBoxom przypisać jednej zmiennej (tutaj x). Ale zauważmy, że program wykonuje zapis do x a tylko raz. Jest on przecież zależny od IF-a.

Instrukcja IF nie oferuje czegoś takiego jak przedziały. Tak np. nie możemy napisać instrukcji IF z przedziałem (np. od 20 do 70). Operujemy tylko operatorami (>,<,=,<=,>=,<>). Taki komfort daje nam za to funkcja Case. Teraz napiszemy ten sam program z użyciem właśnie tej funkcji:

```
Dim wiek As Currency
wiek = InputBox("Podaj swój wiek...", "Pytanie o wiek")
Select Case wiek
Case is < 0 : x=MsgBox("To jest niemożliwe", vbCritical, "Błąd")
Case 0 to 10: x= MsgBox("Mniej niż 11 - jeszcze dziecko", vbInformation,
"Odp")
Case 10 to 18: x = MsgBox("Młodzież", vbInformation, "Odp")
Case 18 to 40: x = MsgBox("Dorosły  do 40", vbInformation, "Odp")
Case 100: x = MsgBox("Piękny wiek (100) !!!", vbInformation, "Odp")
Case is <= 110: x = MsgBox("Pełnia wieku...", vbInformation, "Odp")
Case else: x = MsgBox("Niemożliwe !!!", vbCritical, "Błąd")
End Select
```

Program stał się dużo czytelniejszy.

Opis

1.Dim - wiadomo, nakazujemy programowi pamiętać wiek

2.wiek=InputBox... - Pytanie o wiek, który zostaje zapisany do zmiennej wiek

3.W odróżnieniu od IF-a, Case należy zacząć - podajemy tutaj co będzie wchodziło w rachubę (w naszym przypadku wiek)

4.Jeśli mniejsze niż zero (oczywiście wiek) to MsgBox - "To jest niemożliwe..."

5.Jeśli od 0 do 10 to MsgBox - "Mniej niż..."

6,7 - tak samo

8. Jeśli 100 to MsgBox - "Piękny wiek"

9. Jeśli mniej niż 110 to MsgBox "Pełnia wieku..."

10 Case Else - Jeśli inaczej to MsgBox "Niemożliwe..."

11 End Select - Zakańczamy Case (Odpowiednik End IF)

Uwaga - jeśli używamy operatorów większy - mniejszy itp. to po instrukcji Case trzeba dać słowo is. Jednak jeśli ktoś zapomni napisać tego słowa, VB zrobi to automatycznie. Dwukropek jest odpowiednikiem If-owego Then. Tak więc instrukcja Case jest dużo bardziej uproszczona, chociaż ma większe możliwości niż IF.

Funkcji Select Case używamy wtedy, gdy mamy dużo rozpatrzeń do obsłużenia. Dużo szybciej jest pisać program z Case. Jednak gdy mamy tylko dwie opcje do rozpatrzenia to funkcja IF jest całkowicie wystarczająca.

Przykład

Ćwiczenie: Napisz program pytający użytkownika o wagę (w kg) i wzrost (w cm). Wyliczony współczynnik wg wzoru: (wzrost-100)/waga podziel na sześć kategorii, na które będziesz reagować różnymi komunikatami. Np. jeśli współczynnik będzie minimalnie większy od 1 to możesz wyświetlić komunikat „masz niewielką niedowagę” a jeśli minimalnie mniejszy od 1 to komunikat: „masz niewielką nadwagę”.

Rozwiązanie (kompletne!)

14 Komentarze

Komentarz jest to pomocniczy tekst w projekcie służący jedynie piszącemu program! Visual Basic w ogóle na to nie zwraca uwagi. Po co to stosować... Na razie piszemy małe programy i komentarz jest zbyteczny. Jeśli jednak będzie trzeba wykonać skomplikowany program, będzie można się pogubić w zmiennych i przeznaczeniu poszczególnych pętli. Do deklaracji komentarzy stosujemy znak ' Wszystko co znajdzie się po tym znaku aż do końca linijki jest komentarzem i VB nie zwraca na to najmniejszej uwagi. Komentarze zostają zaznaczone przez VB na zielono.

Przykład:

```
If x=0 then end  'Jeśli zmienna x będzie równa zero to zakończ program
```

lub

```
If x=0 then end
'Jeśli zmienna x będzie równa zero to zakończ program
```

Komentarz ułatwi zrozumienie co w danej linijce robi program. Jest to szczególnie ważne jeśli wrócimy do edycji programu po dłuższej przerwie.

15 Funkcja Val

Jest to funkcja konwertująca tekst na liczbę. Znamy już co prawda funkcję CSng, ale Val jest lepsza. ;)

Różnice pomiędzy Val i CSng prześledź na przykładzie:

Gdyby napisać

```
x = "12"
odp = CSng(x)
```

w zmiennej odp znalazło by się 12. To co jest w polach tekstowych nazywa się stringami. Zapisuje się je właśnie w cudzysłowach. String "12" zostaje przekonwertowany na 12. Jeśli jednak zapiszemy następującą sekwencję:

```
x = "Tata i mama"
odp = CSng(x)
```

to w tym momencie wyskoczy błąd (13). Nie można przekonwertować "Tata i mama" na liczbę. Jednak funkcja VAL to „potrafi”.

```
x = "Tata i mama"
odp = Val(x)
```

teraz w zmiennej odp znajdzie się liczba 0.

Funkcję Val stosujemy w zastępstwie funkcji CSng. Funkcja Val potrafi poradzić sobie z ciągiem tekstowym. Jeśli program oczekiwał liczby przypisze mu wartość zero (0). Zapobiegnie to powstaniu błędu "Type mismath".

Zadanie

Rozbuduj program z rozdziału Funkcja Select Case na stronie ***15***. Uodpornij program na wpisywanie nieprawidłowych danych jako wagi i wzrostu. W tym celu przekształć pobierane dane: waga i wzrost za pomocą funkcji Value (Val). Zwróć uwagę na to, że program wykonuje dzielenie przez zmienną waga zatem nie powinna być równa zero. Jeśli użytkownik wpisze wartość wagi równą zero bądź wstawi tam znak, który nie jest cyfrą, wartość zmiennej będzie równa zero. W takim przypadku należy sprawdzić wartość zmiennej **waga** zanim dokonamy obliczeń, aby działanie programu nie zakończyło się błędem. Można w tym celu użyć funkcji IF. Jeśli waga będzie większa od zera to program powinien wykonać obliczenia BMI i wyświetlić komunikat, jeśli waga będzie mniejsza lub równa zero to powinien wyświetlić się komunikat, że waga została podana nieprawidłowo.

Rozwiązanie (częściowe – tylko to zadanie)

16 Pętle

16.1 Do while

Pętla, jak sama nazwa mówi to kilkukrotne powtórze nie się tych samych instrukcji kodu. Ile ich będzie, to zależy od warunku pętli. Najprościej jest to wyjaśnić na przykładzie. Napiszemy program pytający się nas o wiek. Po jego podaniu wyskoczy komunikat "Masz x lat". Oczywiście program musi być głupotoodporny i jeśli podamy, że mamy 0 lub 123 lata powinien zapytać nas jeszcze raz o wiek.

Utwórz przycisk i dodaj kod:

```
Dim wiek as string
Dim b,c as byte
wiek = InputBox ("Podaj ile masz lat","Pytanie")  'Tutaj program pyta nas o
wiek
If wiek = "" then end    ' Jeśli nacisnęliśmy Cancel
wiek = Val(wiek)  'opis poniższej funkcji pod przykładem, na razie musisz
wiedzieć, że działa podobnie jak CSng( )
Do While wiek<=0 or wiek>120
b = MsgBox ("Musisz podać prawidłowy wiek",vbCritical,"Błąd")
wiek = InputBox ("Podaj jeszcze raz swój wiek","Pytanie")  'Tutaj program
znów pyta nas o wiek
If wiek = "" then end    ' Jeśli nacisnęliśmy Cancel
wiek = Val(Wiek)
Loop
c = MsgBox("Masz " & wiek & " lat" , vbInformation, "Odpowiedź")
```

Objaśnienie działania programu:

1. Dim wiek – deklaracja zmiennej wiek - czyli nasza odpowiedź

2. Dim x,y - deklaracja tego co zwraca MsgBox

3. Pytanie o wiek, który zostaje zapisany do zmiennej wiek

4. If... - jeśli naciśniemy cancel to koniec programu

5. wiek = Val(Wiek) – UWAGA nowość! Jest to funkcja bardzo podobna do CSng( ). W przyszłości właśnie jej będziemy używać.

6. Do while wiek <= 0 or wiek >120 then

Jak widać funkcja jest bardzo podobna do funkcji IF - lecz nie oznacza ona jeśli a dopóki. Czyli tłumacząc - dopóki wiek <= 0 lub wiek >120 wykonuj instrukcje do Loop. Innymi słowy mówiąc - jeśli podamy liczbę taką jak trzeba to program przeskoczy aż za funkcję Loop (Odpowiednik End IF). Jeśli jednak podamy np 140 to...

7. MesageBox, że zły wiek

8.InputBox - ponawiamy pytanie o wiek...

9. Znów sprawdzenie czy naciśnięto Cancel

10. Val - znów konwersja stringu Wiek na liczbę

11. Loop - funkcja która spowrotem przenosi nas do funkcji Do While

No to wracamy, aż do linijki 6 i tu znowu rozstrzyga się czy będziemy już kończyć, czy jeszcze nie. Znów rozpatrzymy dwa przypadki:

a) Znów podaliśmy zły wiek, więc program ponownie wykonuje to co jest między Do While a Loop. Gdy dojdzie do końca to znów wraca do linii 6, i po raz kolejny rozstrzyga sprawę.

b) Podaliśmy dobry wiek - program więc przeskakuje przez funkcję Do While - Loop i lądujemy w linijce 12

12.MsgBox - Program daje nam odpowiedź ile mamy lat.

Przypominam, że zastosowanie znaku & łączy ciągi tekstowe:

Zapis

```
x = "Mam " & 12 & " lat"
```

Daje to w wyniku "Mam 12 lat"

Przykład

Ćwiczenie:

Zmodyfikuj program, który zrobiliśmy ostatnio sprawdzający współczynnik wagi do wzrostu. Jednak niech program pozwoli wpisać wagę tylko z przedziału od 30 do 400kg i wzrost od 110 do 300cm. Wpisanie wielkości spoza dopuszczonych przedziałów powinno wywołać komunikat informacyjny, w jakim przedziale musi się mieścić dopuszczalna waga i wzrost.

Podpowiedź: Musisz zastosować dwie pętle jak w powyższym rozdziale. Jedną przy pytaniu o wzrost (tutaj ewentualny komunikat, że trzeba wpisać wielkość pomiędzy 110 a 300). Drugą przy pytaniu o wagę (tutaj ewentualny komunikat, że trzeba wpisać wielkość pomiędzy 30 a 400).

Proszę zadbać również o szczegóły takie jak poinformowanie użytkownika programu w jakich jednostkach wpisuje swoją wagę i wzrost. Można to zrobić w zapytaniu np. „Wpisz swój wzrost w cm”, „Wpisz swoją wagę w kg”.

Rozwiązanie (kompletne!)

16.2 Do until

Przeciwność funkcji Do While, jest funkcja Do Until. Działa dokłanie odwrotnie. Oto przykład:

Zamiast:

```
Do While wiek<=0 or wiek>120
```

możemy zapisać:

```
Do Until wiek > 0 and wiek <= 120
```

Pamiętaj, że przeciwnością < jest >=, a <= jest >. Przeciwnością <> jest =.

Pamiętaj także o zmianie operatorów - **Or** zamienia się na **And** i odwrotnie.

Linię możemy przetłumaczyć tak:

Dopóki wiek będzie nie większy niż 0 i nie mniejszy niż 120 to...

Funkcji Do Until używa się znacznie rzadziej, ale w niektórych przypadkach użycie jej będzie wygodniejsze niż Do while.

16.3 Pętle odwrotne.

Pętla: Do … While... Loop

Odwrotna: Do... Loop While

Tak samo możemy postąpić z Until. Różnice w stosunku do normalnej pętli najlepiej zobrazować na przykładzie.

```
Blok instrukcji (1)
Do While...
Blok instrukcji (2)
Loop
Blok instrukcji (3)
```

W przypadku normalnej pętli program wykonuje się tak:

Blok instrukcji (1), teraz albo blok instrukcji (2), albo po kilku (lub jednej) pętli blok instrukcji (3)

A teraz funkcja odwrotna do poprzedniej

```
Blok instrukcji (1)
Do
Blok instrukcji (2)
Loop While...
Blok instrukcji (3)
```

A teraz program wykonuje się tak:

Blok instrukcji (1), Blok instrukcji (2), teraz albo blok instrukcji (3), albo znów Blok instrukcji (2) i tak kilka razy (lub raz) i na koniec blok instrukcji (3)

Generalnie różni się to tym, że w funkcji Do While - Loop, program może, ale nie musi wykonąć tego co znajduje się między DO - LOOP. Zaś w funkcji odwrotnej program musi wykonać przynajmniej jeden raz instrukcję zawartą między DO - LOOP.

Funkcja jest rzadko stosowana bo zawsze ją można zastąpić zwykłym ułożeniem. Ale warto wiedzieć, że coś takiego jest.

16.4 Pętla For

Pętla For różni się od pętli Do While, Do Until tym, że wykonuje się określoną ilość razy. Pętle Do – wykonują się dopóki ich warunki nie są spełnione.

Przykład:

```
Dim x,y as byte
For x = 1 to 5
y = MsgBox("To jest MessageBox nr. " & x, vbinformation,"Przykład")
Next x
```

Ten program powoduje pięciokrotne wyświetlenie się MsgBox-a. Jak to działa?

1. Dim - wiadomo co... ;)
2. Dla x = od 1 do 5
3. MsgBox...
4. Dodaj do x i wróć

Analiza:

Najpierw zainicjowanie zmiennych x i y. Dalej jest instrukcja for. Decyduje ona ile razy ma zostać wykonana pętla. W naszym przypadku jeśli x=6 to przeskakuje za Next-a (Odpowiednik Loop-a z pętli DO-LOOP). Czyli tak - obecnie w zmiennej x jest 1 - to jest ta pierwsza liczba w FOR. Wykonuje się msgBox z wiadomością "To jest MessageBox nr 1". Program dochodzi do funkcji Next x - oznacza ona - dodaj (1) i wróć . Tak więc program wraca do For-a, tylko teraz w x jest liczba 2 - i

znów MsgBox - To jest MessageBox nr 2. I znów dodaj 1 i wróć, aż tak do 5. Ale co potem... Załóżmy, że w x jest 5. Jesteśmy przy funkcji For - skoro do pięciu to jeszcze raz pętelka - MessageBox i dochodzimy do Next x. Po tej operacji x = 6. Program wraca do for-a i okazuje się, że x nie spełnia warunku pętli. Następuje przeskok za instrukcję Next - i program w ten sposób kończy działanie.
Dopełnieniem funkcji FOR - jest funkcja Step (czyli krok). Po prostu do funkcji For można dopisać Step i liczbę, o którą ma rosnąć zmienna w pętli. Zastosujmy to do wcześniej napisanego przykładu.

```
Dim x,y as byte
For x = 1 to 5 step 2
y = MsgBox("To jest MessageBox nr. " & x, vbinformation,"Przykład")
Next x
```

Program wchodzi w funkcję FOR, która zapisuje do zmiennej x wartość 1. Dostajemy wiadomość "To jest MessageBox nr. 1". I zatrzymujemy się na next. Funkcja STEP nakazuje FOR-owi, aby teraz do x dodał 2. W x znajduje się liczba 3! Teraz pojawi się MessageBox z numerem 3. W końcu x osiągnie wartość= 7, gdzie nastąpi przeskok za funkcję Next i koniec programu.

Teraz zastosujmy funkcję for do liczenia silni danej liczby.

Co to jest silnia:7! (czytaj. 7 silnia) = 1*2*3*4*5*6*7

Na formie ułóż przycisk - jako wartość "Caption" wpisz Silnia, i nazwij go cmdSilnia. Zrób textBox-a. Wyczyść wartość Caption i nazwij go txtSilnia. Teraz dodaj do przycisku następujący kod:

```
Dim x, odp
odp = 1
For x = 1 To txtSilnia
odp = odp * x
Next x
ss = MsgBox(txtSilnia & " ! to " & odp)
```

Analiza:

1. Dim - zainicjowanie zmiennych
2. 2. Na początek ustalenie zmiennej początkowej.
3. 3. Instrukcja for - od 1 do... ile? txtsilnia to jest właśnie liczba którą wpisałeś w okienko.
4. 4. Teraz następuje przemnożenie stałej przez x - czyli kolejną liczbę naturalną zaczynając od 1, a kończąc na txtSilnia. Czyli nasze równanie (załóżmy, że w okienko wpisałeś 7) 1*2*3*4*5*6*7 czyli 7!. Po 7 okrążeniach otrzymujemy nasz wynik.

Realizacja pętli for plus ćwiczenie

Ćwiczenie 1:

Napisz program z przyciskiem i polem tekstowym. Program ma dodawać tylko liczby parzyste zaczynając od 2, a kończąc na liczbie wpisanej w pole textowe.

Podpowiedzi: Dodawanie zaczynamy od 2 (bo 1 jest liczbą nieparzystą) funkcję FOR też zaczniemy od 2. Musimy użyć funkcji STEP.

Ćwiczenie 2

Napisz program sumujący wielokrotności podanej liczby w zakresie od zera (0) do wartości maksymalnej podanej przez użytkownika.

Rozwiązanie

17 Zmiana właściwości obiektów (komponentów) kodem.

Dla przypomnienia

(name)	W tym polu wpisujemy nazwę identyfikującą naszą etykietę – najlepiej używać skrótu lbl (od Label) np. lblNazwa
TextAlign (Alignment)	!!! Czyli wyrównanie tekstu który znajduje się na etykiecie. Domyślnie mamy ustalone 0 - Left Justyfy - czyli wyrównanie do lewej krawędzi. Można też ustawić 1 - Right... czyli wyrównanie do prawej oraz 2 - Center - czyli wyśrodkowanie tekstu.
Caption	Chyba najważniejsza – Tu ustalamy wyrazy które mają znaleźć się na naszej etykiecie
Font	W tym oknie możemy ustalić czcionkę, wielkość i efekty dla napisu
Height	Wysokość etykiety. Raczej wygodniejsze jest rozciąganie bezpośrednio na formularzu
Left	Określa odległość etykiety od lewej krawędzi ekranu. Wygodniej jest przemieszczać etykietę bezpośrednio na formie.
BackColor	Tutaj kolor tła etykiety
ForeColor	Tutaj kolor tekstu
Top	Określa położenie etykiety od górnej krawędzi ekranu
Width	Określa szerokość etykiety
Visible	Określa czy etykieta ma być widoczna na formularzu. Jeśli zaznaczymy false, to po uruchomieniu programu zobaczymy pusty formularz.

Napiszmy mały program. Na formularzu umieśćmy pole tekstowe o nazwie txtWpis, wyczyść opcję Caption. Umieszczamy także etykietę o nazwie lblGotowe i także wyczyśćmy Caption. I jeszcze 3 przyciski o nazwach - Do lewej, Do prawej, Wyśrodkowanie. Nasz program ma działać tak - Wpisujemy w pole tekstowe jakiś tekst, który automatycznie przenoszony jest do etykiety. A nasze trzy przyciski sterują położeniem tekstu na etykiecie.

Trzeba pamiętać, że chcemy przenieść tekst z pola tekstowego do etykiety. Przyjmijmy, że program ma zareagować wtedy, gdy zaczniemy coś wpisywać do pola tekstowego. Kod zatem musi być przypisany do pola tekstowego. Trzeba kliknąć w pole dwukrotnie i widzimy coś takiego:

```
Private Sub txtwpis_Change()

End Sub
```

Change - oznacza zmianę - czyli jeśli coś będziemy zmieniać w polu tekstowym to wykona nam się kod. Chodzi nam o przeniesienie tego co w txtWpis do lblGotowe. Wystarczy napisać:

```
lblGotowe.caption = txtwpis
```

teraz ustawiamy przyciski - klikamy dwukrotnie przycisk Do lewej i dodaj kod:

```
lblGotowe.TextAlign= 0
```

Jak wiadomo z wcześniejszej tabelki 0 oznacza do lewej:

```
etykietaGotowe.wyrównanie = 0
```

czyli do lewej

Tak samo należy dopisać kod do pozostałych przycisków. Popatrz do tabelki i dopisz odpowiedni kod.

Zapisz program - będzie jeszcze potrzebny.

Tym sposobem można zmieniać właściwości wszystkich elementów ułożonych na formie!!!

Powyższy Przykład

Ćwiczenie: Napisz program, w którym będziesz umiał na UserFormie przemieszczać textboxem. Niech pozwalają na to przyciski Góra, Dół, Lewa, Prawa .

Przykład wykonania przemieszczania obiektu po userformie.

Parametr Visible, czyli znikanie obiektów - przykład

Sprawdzian po tej lekcji – zadania obliczeniowe

18 Pola wyboru

18.1 Check Box

Pola wyboru to: Check Box i Option Button (Radio). Znajdują się one na palecie narzędzi w czwartym rzędzie. Taki kwadracik „zafajkowany” i kółeczko z kropką. Na początek wstawimy ten pierwszy czyli kwadracik.

Służy on do poinformowania programu czy coś chcemy czy też nie. Np. możemy nim sterować podkreślenie tekstu w etykiecie. Wykonajmy przykład:

Na formie ułóż CheckBox-a. Nazwij go chkPodkreslenie. Właściwość Caption ustaw na „Podkreśl”. Podkreślany będzie jakiś tekst w etykiecie. Tworzymy więc etykietę o nazwie lblTekst i tekście (Caption) – „To jest jakiś tam tekst”. OK. Teraz nasz program po zaznaczeniu Check-a powinien podkreślić tekst. Oczywiście kod będziemy dodawać do Check-a. Klikamy więc go dwukrotnie i dopisujemy kod:

```
lblTekst.FontUnderline = True
```

właściwość FontUnderline – służy do podkreślenia tekstu. True to prawda, czyli włączenie funkcji podkreślenia dla tekstu.

Niestety nie przewidzieliśmy wyłączenia podkreślenia. Możemy sprawdzić czy Check Box jest zaznaczony czy też nie? Steruje tym funkcja Value. Jeśli jest ona równa 0 to Check jest niezaznaczony, jeśli 1 to zaznaczony, a jeśli 2 to zszarzały (niedostępny). Żeby zmieniać tę wartość należy sprawdzić Check i w oknie properties zmienić kod właściwości.

Cała procedura powinna wyglądać tak:

```
If chkPodkreslenie.Value = True Then
lblTekst.FontUnderline = True
Else
lblTekst.FontUnderline = False
End If
```

Przetłumaczmy.

Jeśli Check Box zostanie zaznaczony (bo Value będzie równe True) to Podkreśl tekst. W przeciwnym przypadku (czyli wyłączymy Checka – Value = False) Skasuj podkreślenie tekstu. Koniec

Przykład

Ćwiczenie: Zmodyfikuj program służący do sprawdzania współczynnika wagi do wzrostu. Program ten wykonywaliśmy na wcześniejszych zajęciach. Dotychczas program umożliwiał sprawdzenie współczynnika i pokazywał odpowiedni komunikat w zależności od tego jaki był stosunek wzrostu człowieka do jego wagi. Teraz przebuduj program tak aby:

1. komunikat pojawiał się w polu Label na UserForm’ie. Dodatkowo pole Label z tym komunikatem powinno zmieniać swoje położenie. Czym wpiszemy większy wzrost tym pole Label powinno pojawiać się wyżej na UserForm’ie. Parametr – Top.
2. wielkość czcionki pola Label powinna się zmieniać wraz z wpisaną w program wagą. Czym ktoś będzie cięższy tym większą czcionką niech będzie napisany komunikat. Parametr – Font.
3. Niech na UserFormie pojawi się CheckBox, który będzie umożliwiał włączenie bądź wyłączenie zmiany właściwości obiektów w zależności od wyników z powyżej opisanych opcji 1 i 2.

Podpowiedź w zakresie odwzorowania wzrostu i wagi na userformie przy pomocy pola label. Graficzna prezentacja wagi i wzrostu - przykład.

18.2 OptionButton (RadioButton).

Zobacz przykład działania OptionButton'a i porównaj z CheckBox'em.

Aby się o tym przekonać czym różni się CheckButton od OptionButton'a umieść na formie dwa Check-i i dwa OptionButtony. Uruchom teraz program. Najpierw zaznacz pierwszego CheckBox-a a potem drugiego. To samo zrób na OptionButton'ach. Jak widzisz nie da się zaznaczyć naraz dwóch OptionButton'ów. Sama nazwa Option (opcje) wskazuje, że będziemy tutaj wybierali jedną z opcji.

Przykład: programik pokazujący praktyczne zastosowanie obu tych pól wyboru. Napisz małą ankietę

Na formularzu umieść dwa OptionButton-y jako Caption ponadawaj im kolejno etykiety – Pentium i AMD oraz nazwij je optPentium, optAMD. Stwórz także jeden CheckBox i jako właściwość Caption nadaj mu etykietę Posiadam akcelerator graficzny i nazwij go chkAkcelerator. Potrzebna nam także będzie etykieta. Skasuj zawartość Caption i nazwij ją lblEtykieta. Nasz program ma działać tak: W etykiecie ma się pojawić tekst – Masz komputer z procesorem (nazwa) i (masz/niemasz) akcelerator(a). I tu trzeba wiedzieć, że OptionButton-em można sterować za pomocą opcji Value (wartość) tylko z tą różnicą, że nie ma tu już wartośći 0,1,2 tylko jest True lub False. Czyli klikamy dwukrotnie na pierwszym z optionButtonów i dodajemy następujący kod:

```
Dim procesor, akcelerator As String
If optPentium.Value = True Then
procesor = "Pentium"
Else
procesor = "AMD"
End If
If chkAkcelerator.Value = True Then
akcelerator = "i posiadasz akcelerator graficzny"
Else
akcelerator = "bez akceleratora graficznego"
End If
lblEtykieta = "Masz komputer z procesorem " & procesor & " " & akcelerator
```

Powyższy przykład z rozwinięciem o napęd optyczny.

Wszystko byłoby w porządku gdyby nie trzeba było napisać wiele razy tego samego kodu (do każdego option buttona i checz box'a).

W tym wypadku możemy wykorzystać funkcję Call. Oznacza ona Połącz.

19 Funkcja Call

Zasada działania funkcji Call:

```
Linijka1
Linijka2
Call Podprogram
Linijka3
Linijka4

Private Sub Podprogram( )
Linijka5
Linijka6
Linijka7
End Sub
```

Program będzie się wykonywał tak – Pierwsze linijka 1 i 2. Następnie napotykamy Call – łączy nas z podprogramem – czyli następnie linijka 5,6,7 i do End Sub. W tym momencie program wraca do Call czyli teraz linijki 3 i 4.

Jak to zastosować do naszego programu:

Najpierw musisz skasować wszystko co napisałeś do tej pory, żeby się nie pogubić.

Teraz musisz napisać procedurę, która będzie wywoływana Call-em. W oknie kodu dopisujemy ją na samym końcu – zawsze za ostatnim End Sub.

Napisz:

```
Private Sub Przelicz ( )
```

Po naciśnięci Enter VB dopisał End Sub i oddzielił procedurę od reszty. Piszemy dalej to co wpisywaliśmy wcześniej do każdego z przycisków opcji czyli:

```
Dim procesor, akcelerator As String
If optPentium.Value = True Then
procesor = "Pentium"
Else
procesor = "AMD"
End If
If chkAkcelerator.Value = True Then
akcelerator = "i posiadasz akcelerator graficzny"
Else
akcelerator = "bez akceleratora graficznego"
End If
lblEtykieta = "Masz komputer z procesorem " & procesor & " " & akcelerator
```

Aby program „wiedział”, że po zaznaczeniu jednej z opcji ma się odwołać do tej procedury, w każdym przycisku opcji dopisujemy linijkę:

```
Call Przelicz ( )
```

Czyli po naciśnięciu każdego przycisku wywołamy tę samą procedurę o nazwie przelicz.

Przykład

Zadanie: Napisz program sprawdzający współczynnik (wzrost/waga) i zamieść na nim dwa option buttony.

pierwszy niech pozwoli na określenie czy waga była mierzona z ubraniami czy bez ubrań.

drugi niech pozwoli wybrać czy wzrost był mierzony z kapeluszem czy bez kapelusza.

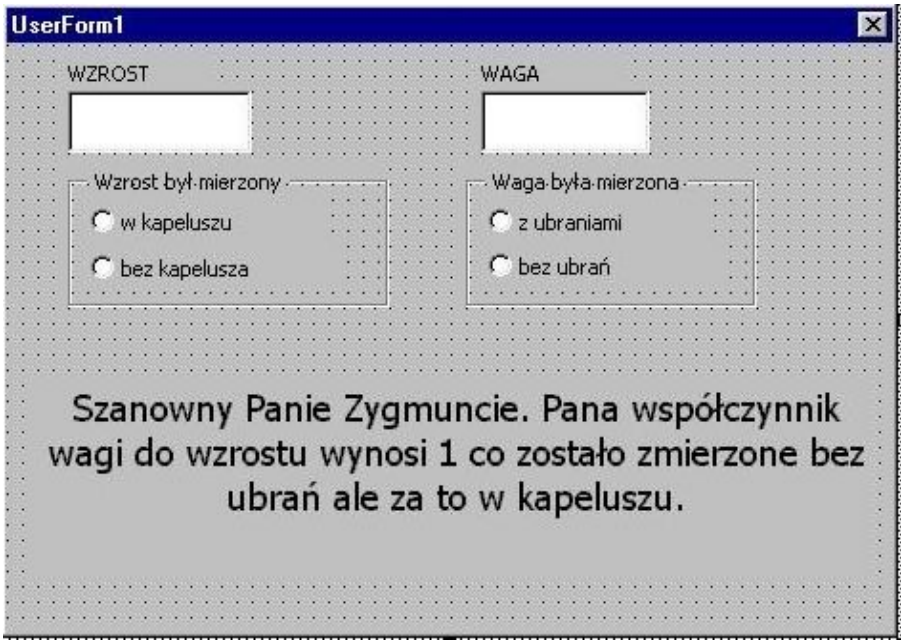
Zapytaj też o imię osoby, która wpisuje swoje dane.

Komunikat, który będzie pojawiał się na końcu niech poda wartość współczynnika wzrostu do wagi a także doda komentarz czy waga została podana z ubraniami czy bez oraz czy wzrost był mierzony w kapeluszu czy bez niego. Przykładowy komunikat powinien wyglądać tak:

„Szanowny Panie X. Pana współczynnik wynosi Y co zostało zmierzone bez ubrań ale za to w kapeluszu”

Podpowiedź: żeby odseparować od sioebie OptionButtony dotyczące wagi od wzrostu należy umieścić je w osobnych ramkach (Frame). Patrz rysunek poniżej.

Okno programu może wyglądać np. tak:



Rozwiązanie

20 Frame, czyli ramka

Przypomnijmy sobie program z poprzednich lekcji. Pisaliśmy program, gdzie OptionButtony decydowały o naszym procesorze (a CheckBox - czy mamy akcelerator graficzny). A jak byśmy jeszcze chcieli dorobić np. typ napędu optycznego? Konieczne byłoby zaznaczenie dwóch OptionButtonów. Jednego decydującego o procesorze, a drugiego o napędzie. Jak wiadomo nie można zaznaczyć więcej niż jednego pola option button w obrębie user forma.

Do tego celu musimy użyć narzędzia o nazwie Frame (Ramka). Jest ono na palecie narzędzi w postaci szarego kwadracika z małym napisem xy u góry. Po umieszczeniu tego obiektu na formularzu możemy w jego obrębie umieścić niezależną grupę OptionButton'ów. Nie będą one kolidowały z OptB z sąsiednich ramek, czy z userforma.

Czyli więcej grup Option Butonów możemy stworzyć

1. OptBttn bezpośrednio na formie + OptBttn w ramce (frame)
2. OptBttn w ramce1 + OptBttn w ramce2 itd..

Oczywiście grup można tworzyć więcej. Trzeba tylko pamiętać o ich rozdzieleniu za pomocą ramek.

UWAGA!!!
W wersji VB Professional Najpierw tworzymy ramkę, a potem przycisk opcji tworzymy bezpośrednio na niej. Nie możemy przenosić OptB-ów na ramkę. Jest to natomiast możliwe w VB for Applications.

ZADANIE DOMOWE

Udoskonal program z poprzedniej lekcji, tak aby można było zaznaczyć typ napędu optycznego.

Komunikat wynikowy powinien być w następującej formie:

Masz komputer z procesorem {AMD, Pentium}, napędem optycznym (DVD, CD-ROM, CD-RW, DVD-RW) i (posiadasz akcelerator, nie posiadasz akceleratora).

21 Zmienne globalne (global)

Do zadeklarowania zmiennej służy funkcja DIM. Tak zadeklarowana zmienna jest „pamiętana” do zastosowania End Sub - czyli zakończenia procedury. Jeśli chcemy aby wartość zmiennej została zapamiętana dłużej to musimy użyć funkcji GLOBAL i umieścić ją w innym miejscu naszego programu.

Przykład. Porównanie użycia GLOBAL i DIM.

Na formie ułóż 2 przyciski - jako etykiety nadaj im **Zapamiętaj** i **Odczytaj** oraz jeden TextBox (Pole tekstowe) nadaj mu nazwę txtzmienna. Po naciśnięci przycisku Zapamiętaj nasz program ma zapisać do zmiennej **x** to co będzie w TextBox-ie. Przycisk Odczytaj ma spowodować wyświetlenie się Okna komunikatu które wyświetli wartość zmiennej **x**.

Do przycisku **Zapamiętaj** dodajemy kod:

```
Dim x as string
x = txtzmienna   ' zapisujemy do x to co jest w TextBox-ie
```

A do przycisku Odczytaj:

```
Dim odp
odp = MsgBox (x)  ‘wywołujemy message boxa z wypisaną wartością x
```

Uruchom program. Najpierw naciśnij Zapamiętaj, a potem Odczytaj.

Wyświetla się nam pusty komunikat.

Przeanalizujmy program...

Po naciśnięciu **Zapamiętaj** program przypisuje do **x** to co jest wpisane do textboxie. Następnie mamy zakończenie procedury, czyli End Sub. Po wciśnięciu przycisku **Odczytaj** wykonuje się nowa procedura, która ma wyświetlić wartość **x.** Jednak **x** zostało już „zapomniane” przez program i dlatego otrzymujemy puste okienko.

Aby zmusić program, do pamiętania **x** przez cały czas należy skorzystać z funkcji GLOBAL. Należy ją załadować wcześniej. Oprócz form w programie są jeszcze moduły - one ładują się najwcześniej. Do projektu dodamy zatem moduł.

W oknie Project kliknij prawym i wybierz ***Insert > Module***. Z prawej strony pojawiło się okienko tekstowe z modułem. Można je w każdej chwili zamknąć i ponownie otworzyć za pomocą dwukliku w oknie **Project**.

W module dopisz taką linijkę:

```
Global x as string
```

Funkcja działa identycznie jak DIM, z tą różnicą, że jest "długowieczna". Musimy jeszcze usunąć funkcję Dim x as string z jednego z przycisków, żeby nie deklarować zmiennej dwa razy. Przykład

22 Okna dialogowe (Common Dialog Box)

22.1 Czcionka i kolor

Okna dialogowe to gotowe okna (userformy) z przygotowanymi obiektami do wykonywania typowych zadań w systemie operacyjnym.. Można je zobaczyć np. w Paincie. Uruchom Paint-a (Start - Programy - Akcesoria - Paint). Gdy się uruchomi to z menu wybierz plik i otwórz - pojawiło się okno dialogowe. Takie gotowe okna są jeszcze dla funkcji **zapisz** - drugie okno dialogowe bardzo podobne do otwórz. Na dole jest paleta z farbami - kliknij dwukrotnie na jakimkolwiek kolorze. Wyskoczyło kolejne okno dialogowe - kolor. Są jeszcze okna dialogowe - czcionka i drukuj. W przykładzie wykorzystamy dwa - kolor i czcionka.

Wcześniej pisaliśmy program, w którym formatowaliśmy tekst. Potrafił wyrównywać tekst do prawej, lewej i wyśrodkowywać. Teraz uzupełnimy go o możliwość manipulowania czcionką tzn. wielkość, styl i rodzaj czcionki. Uzupełnimy go także o możliwość zmiany koloru czcionki. Dodaj do UserForma dwa przyciski. Jako wartość caption przypisz im kolejno - **Czcionka**, **Kolor**.

Teraz na formie trzeba ułożyć element, który nazywa się Common Dialog Box. W VB for Applications nie występuje w domyślnym przyborniku (Toolbox). Można go dodać przez menu Tools > Additional Controls lub przez kliknięcie prawym klawiszem myszki na wolnym miejscu na pasku narzędzi (Toolbox). Należy wybrać ***Microsoft Common Dialog Control 6.0***

Ten obiekt należy ustawić gdzieś na UserFormie, obojętne gdzie - on i tak jest niewidoczny. Potrzebny jest on VB tylko po to aby zapamiętywał własności wybrane z okien dialogowych. Zmień jego nazwę na cdb1. Do przycisku **Czcionka** dopisz kod:

```
cdb1.DialogTitle = "Zmień
czcionkę"
```
tytuł okna dialogowego

```
cdb1.FontName = "MS Sans
Serif"
```
ustawiamy domyślną czcionkę

```
cdb1.Flags = cdlCFBoth
```
Informujemy CDB-a jakie czcionki ma wyświetlić - cdlCFBoth oznacza, że wszystkie

```
cdb1.ShowFont
```
pokaż okno dialogowe.

Teraz musimy wartości ustawione w oknie dialogowym włączyć w polu label. Dopisujemy zatem następujący kod (zakładam, że pole label nazywa się ***lblgotowe***):

```
lblgotowe.Font = cdb1.FontName
lblgotowe.FontBold = cdb1.FontBold
lblgotowe.FontItalic = cdb1.FontItalic
lblgotowe.FontSize = cdb1.FontSize
```

Do drugiego przycisku (**Kolor**) dodajemy następujący kod:

```
cdb1.dialogTitle = "Zmieniamy kolor"
cdb1.ShowColor
```

i żeby zmiany zostały wyświetlone w polu label potrzebny jest jeszcze wiersz:

```
lblGotowe.ForeColor = cdb1.color
```

Teraz program powinien działać.

Przykład działania Common Dialog Box (działa w Office od wersji 2003). Dodatkowo utrudnienia (brak dostępu do kontrolki) dla uczniów przy pracy w domenie w pracowni szkolnej.

22.2 Okno otwórz

Aby wywołać okno **otwórz** należy posłużyć się następującym kodem:

```
cdb1.dialogTitle = "Tytuł"
cdb1.FileName = "*.*"
cdb1.Filter = "Zwykły plik tekstowe|*.txt|Zdjęcia|*.bmp; *.jpg "
cdb1.showOpen
```

W drugim wierszu filtrujemy wyświetlane pliki. Znaczek | znajduje się nad Enter-em obok Backspace-a. Wpisujemy go z Shift-em. W powyższym przykładzie program będzie tylko widział pliki txt, a po zmianie opcji także bmp i jpg. Jeśli chcesz aby program widział wszystkie pliki to wpisz

```
cdb1.Filter = "*.*"
```

Okno to zwraca ścieżkę do pliku - możesz ją wyciągnąć w ten sposób:

```
Dim x
x = cdb1.fileName
```

22.3 Okno zapisz

```
cdb1.DialogTitle = "Tytuł"
cdb1.Filter = "*.*"
cdb1.FileName = "Test.txt"    'Domyślna nazwa pliku
cdb1.ShowSave
```

Ta funkcja także zwraca właściwość FileName

22.4 Okno drukuj

```
cdb1.DialogTitle = "Tytuł"
cdb.ShowPrinter
```

Ta funkcja zwraca pewne ustawienia drukarki.

23 Lista wyboru (ListBox)

Lista wyboru to takie "okienko", w którym możemy przygotować wiele opcji, z których użytkownik będzie mógł wybrać. Jeśli przygotujemy nasz program odpowiednio to będzie mógł także dodawać i odejmować elementy z listy wyboru.

23.1 Dodawanie elementu do listy

Przykład. Uruchamiamy VB i na formie układamy element nazwany ListBox (ikonka z prawym paskiem przewijania i białym polem). Na user formie wygląda jak TextBox. ListBox nazwij Lista teraz dodaj przycisk na formie i nazwij go cmdDodaj. Jako właściwość Caption nadaj mu Dodaj. Teraz na formie ułóż jeszcze TextBox-a - nazwij go txtWprowadz.

Gdy wpiszemy coś w pole tekstowe i klikniemy Dodaj, to zawartość tego pola tekstowego ma zostać przeniesiona do Listy. Teraz weźmy się za kod. Do przycisku dopisz:

```
Lista.AddItem txtWprowadz     'dodanie elementu do listy
txtWprowadz = ""        'wyczyszczenie pola tekstowego
```

Uruchom program - wpisz coś w pole tekstowe i kliknij Dodaj. Tekst powinien przenieść się do listy! Wpisz coś drugi raz i kliknij Dodaj. Powinna pojawić się druga pozycja.

Przykład

23.2 Usuwanie elementu z listy

Dodajmy do naszego projektu jeszcze jeden przycisk - nazwij go cmdUsun i jako etykietę nadaj mu Usuń. Teraz jeszcze jeden textBox - nazwij go txtUsun. Teraz nasz program będzie jeszcze potrafił usunąć pozycję z listy o podanym numerze. Do przycisku Usuń dodaj następujący kod:

```
Dim x
x = Val(txtusun) - 1
Lista.RemoveItem (x)
```

Uruchom program. Dopisz kilka elementów do listy. Teraz w pole tekstowe **txtusun** wpisz liczbę, która będzie pozycją danej, którą chcesz usunąć z listy. Kliknij Usuń. Działa?

Tłumaczenie: Ostatnia linijka usuwa element x listy. Ale dlaczego wcześniej jest x = to co w polu tekstowym - 1 ? Dla Visual Basica pierwsza pozycja na liście ma wartość 0, druga 1 itd. Aby to unormować odejmujemy jeden. Jeśli w okienko tekstowe wpiszesz 1 program odejmie od tego 1 i wyjdzie mu zero - więc usunie pierwszą pozycję listy. Przypominam, że funkcja VAL ( ) zamienia stringi (dane tekstowe) na liczby.

23.3 Czyszczenie listy

Czyszczenie listy - jeśli chcesz usunąć wszystkie elementy list do jakiegoś nowego przycisku dodaj kod:

```
Lista.Clear
```

23.4 Zliczanie elementów na liście

Do zliczania elementów na liście wystarczy posłużyć się kodem:

```
Dim x
x = Lista.ListCount
```

23.5 Podstawianie pod zmienne elementów z listy

Zostało nam jeszcze podstawianie pod zmienne. Jeśli chcesz podstawić pod zmienną x np. 4 element listy posłuż się kodem

```
Dim x
x = Lista.List(3)
```

Przypominam, że VB zaczyna liczyć od 0. Dla nas 4 element listy to dla niego trzeci.

23.6 Właściwości listy

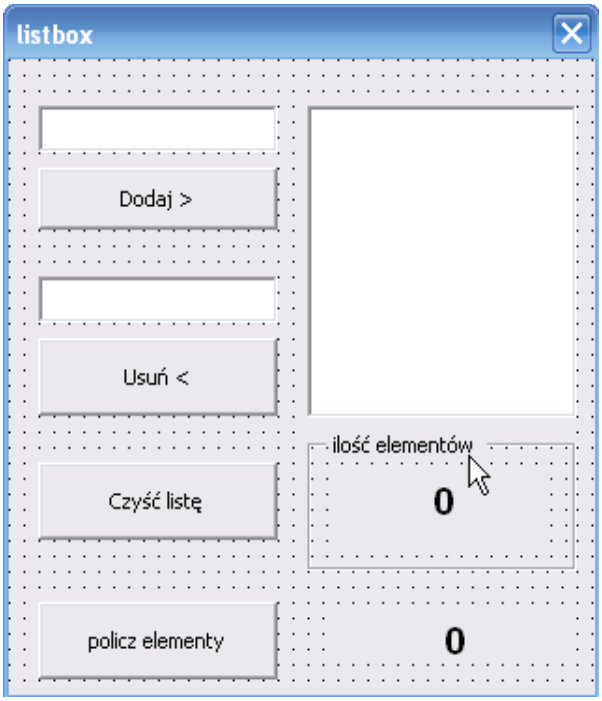
Niektóre właściwości listy.

Columns	Ta właściwość określa ilość kolumn. Jeśli jej wartość jest ustawiona na 0 to mamy zwykłą listę z pionowym paskiem przewijania. Jeśli damy 1 to elementy na liście będą wyświetlone w poziomie. Jeśli damy więcej to elementy będą się układać poziomo, na ekranie będzie widać n kolumn (n to liczba > 1). Uff...
MultiSelect	0 - None - oznacza, że możemy wybrać tylko jeden element listy (podobnie jak było z OptionsButton), jeśli 1- Simple, użytkownik może wybrać kilka elementów klikając na nich. 2 - Extended - użytkownik może wybrać kilka elementów listy pod warunkiem, że przytrzyma klawisz Shift
Sorted	Określa czy lista ma być posortowana alfabetycznie, czy też nie
Style	Określa, czy lista ma wyglądać "normalnie" czy bardziej jak zbiór CheckBox-ów ;)

Zadanie: Wykonaj program, który będzie umożliwiał wstawianie i usuwanie elementów z listy tak jak to zostało opisane w powyższym dziale. Dodatkowo program ma umożliwiać czyszczenie listy i wyświetlenie liczby ilości elementów na liście.

Jeśli to już zrobiłeś to rozbuduj program o automatyczne obliczanie ilości elementów na liście po każdej operacji dodawania i usuwania (czy czyszczenia) listy. Do wykonania tego zadania będziesz potrzebował skorzystać z wcześniej poznanej funkcji GLOBAL. Przyda się także funkcja CALL, aby nie wpisywać tego samego kodu kilka razy.

Okno programu może wyglądać następująco:



Rozwiązanie

24 ComboBox

ComboBox ‘em posługiwać się będziemy identycznie jak ListBox-em. Są tylko dwie różnice. W ListBox-ie mogliśmy ustawić czy będzie możliwość wybrania jednej czy wielu pozycji. W Combo możliwe jest tylko wybranie jednej.

Druga - chyba najważniejsza - zamiast stosować ListBox-a + TextBoxa możemy zastosować sam ComboBox. Musimy posłużyć się funkcją rozpoznającą wciśnięty klawisz.

Przykład

Ułóż na formie ComboBox-a. Kliknij na niego dwukrotnie. Zobacz na górę okna - pisze coś takiego :

```
Private Sub Combo1_Click()
```

My zamiast click będziemy musieli posłużyć się KeyDown. Zmień to Click na KeyDown i dopisz następujący kod:

```
If KeyCode = vbKeyReturn Then
Combo1.AddItem Combo1.Text
Combo1.Text = ""
End If
```

Tłumaczę:

```
Jeśli nacisnąłeś Enter to
Dodaj do list Combo to co wpisaliśmy
Wyczyść to co wpisaliśmy
Koniec Jeśli
```

przykład

Uwaga!

Zwróć uwagę na to, że jeśli chcesz dopisywać do Combo pozycje jedna po drugiej to wygodnie jest zmodyfikować właściwości TabStop obiektów na userformie. Jeśli chcesz, żeby kursor zawsze wracał na pole ComboBox i nie przeskakiwał na inne obiekty po dodaniu wpisu, to zmień ustawienia TabStop wszystkich obiektów, poza polem Combo, na False.

Zadanie. Wykonaj program, który wpisze na listę w ListBox i combobox wielokrotności liczby (x20) wpisanej do textboxa przez użytkownika.

25 Tablice danych

Tablica danych to lista zmiennych tego samego typu (Np. nie możemy pomieszać w tablicy stringów i liczb). Praktyczne zastosowanie tablicy danych:

Przypuśćmy, że musimy napisać program sumujący 20 zmiennych. Wykorzystując już zdobytą wiedzę napisalibyśmy tak:

Dim Liczba1, Liczba2, Liczba3, Liczba4, Liczba5, Liczba6, Liczba7, Liczba8 (itd...)as Single
wynik = Liczba1 + Liczba2 + Liczba3 + Liczba4 + Liczba5 + Liczba6 + Liczba7 + itd...

Sporo pisania. Gdybyśmy jednak zastosowali tablicę danych wyglądało by to tak:

```
Dim tablicaliczb(20)
Dim x as byte
For x = 1 to 20
wynik = wynik + tablicaLiczb(x)
Next x
```

przykład

Prawda, że krócej. Tak samo trwa zapisywanie do zmiennych. Gdybyśmy np. chcieli odczytać dane z pliku to musielibyśmy to robić tak:

```
Dim Liczba1, Liczba2, Liczba3, Liczba4, Liczba5, Liczba6, Liczba7, Liczba8
(itd...) as Single
--- odczyt z pliku - zapisz do zmiennej Liczba1 ---
--- odczyt z pliku - zapisz do zmiennej Liczba2 ---
--- odczyt z pliku - zapisz do zmiennej Liczba3 ---
--- odczyt z pliku - zapisz do zmiennej Liczba4 ---
itd...
```

Gdybyśmy np. musieli odczytać 300 danych z pliku to byłaby to operacja wymagająca wiele linii kodu. Za pomocą tablic danych wykonamy tą czynność w kilku linijkach.

Sposób:

```
Dim tablicadanych(100)
Dim x as Single
For x=1 to 100
--- odczyt z pliku - zapis do zmiennej tablicadanych(x)
Next x
```

25.1 Zadania do działu tablice danych

Zadanie 1

Wykonaj program, który pobierze do tablicy pięć liczb. Na userformie ustaw dwa przyciski, które będą:

Pierwszy – sumowały,

Drugi – mnożyły,

wszystkie liczby zawarte w tablicy. Wynik ma zostać wyświetlony w polu label.

Zadanie 2

Wykonaj program, który umożliwi wpisanie na listę do (maksymalnie) dziesięciu liczb a następnie umożliwi ich podsumowanie. Podpowiedź: skorzystaj z ListBox'a lub ComboBox'a, tablicy danych o dziesięciu elementach i funkcji przypisywania do zmiennej wartości pobranej z listy. Program ma pobierać maksymalnie 10 liczb!

Przykład

Zadanie 3

Wpisz do tablicy 10 liczb. Możesz to zrobić dowolną metodą:

- poprzez nadanie wartości z góry w programie,

- poprzez losowe generowanie wartości,

- poprzez przyjęcie wartości od użytkownika.

Następnie napisz procedurę, która znajdzie w tablicy element największy lub/oraz najmniejszy.

Zadanie 4

Wykonaj program, który wpisze do tablicy danych dziesięć wielokrotności liczby podanej przez użytkownika. Dodaj możliwość wyświetlenia wszystkich elementów tablicy w listbox'ie.

Zadanie 5

Wykonaj program, który przyjmie kilkuelementową tablicę wartości (w dowolny sposób). Po wybraniu przez użytkownika działania

- Mnożenie
- Dzielenie
- Odejmowanie
- Dodawanie

i podaniu liczby program powinien zmienić wszystkie elementy tablicy wg klucza działanie > element. Program po każdym działaniu powinien wyświetlić wartości wszystkich elementów z tablicy (w dowolny sposób).

Przykład:

Została zdefiniowana tablica pięcioelementowa o wartościach (7, 5, 8, 3, 11). Użytkownik wybrał działanie **odejmowanie** i wpisał liczbę **4**. Program powinien zmienić elementy w tablicy odejmując od nich liczbę **4**. Zatem zawartość tablicy powinna być następująca (3, 1, 4, -1, 7).

Sprawdzian z zakresu listbox, call, global, frame (option button), tablice

Sprawdzian teoretyczny na koniec semestru

26 ScrollBar

Ikonka tego obiektu ma pionowy pasek ze strzałeczkami na końcach. Umieść go na formie. Rozciągnij go aby był jak najwyższy. Teraz uruchom program. Wygląda jak zwykły pasek do przewijania tekstu. Faktycznie, tak samo wygląda, ale służy do czego innego. Zwraca on wartość która może posłużyć np. do manipulacji wielkości czcionki w jakiejś etykiecie. Najważniejsze właściwości ScrollBar-a to:

SmallChange	Tłumacząc - Mała zmiana. Oznacza wartość o którą zmieni się ScrollBar po naciśnięciu jednej ze strzałek
LargeChange	Tłumacząc - Duża zmiana. Oznacza wartość o którą zmieni się ScrollBar po kliknięciu gdziekolwiek na pasku
Min	Wartość minimalna paska
Max	Wartość maksymalna paska

Na formie wstaw jeszcze dość dużą etykietę. Nazwij ją lblCzcionka jako wartość Caption nadaj jej TEST. Teraz spowodujemy, aby regulacja ScrollBar-em zmieniała wielkość Czcionki. Musimy zmienić właściwości Min i Max ScrollBara na takie, jakiej wielkości czcionkę chcemy oglądać przy ustawieniach minimalnych i maksymalnych. Następnie należy ustalić jeszcze SmallChange i LargeChange. To oczywiście ilość punktów, o które będzie się zmieniać wielkość czcionki.

Aby program zmieniał wielkość czcionki w zależności od operacji na scrollbarze musimy powiązać te dwa obiekty kodem.

```
lblCzcionka.Font.Size = ScrollBar1.Value
```

Uruchom teraz program i sprawdź jego działanie.

przykład

Sprawdzian z tablic

27 Operacje na dacie i czasie

Visual Basic potrafi wykonywać wiele operacji związanych z aktualnym czasem (datą) czy sprawdzać czas lub datę.

Przykład

Zegar czasu rzeczywistego

27.1 Date

Stwórzmy mały programik, ułóżmy przycisk i dodajmy do niego kod:

```
x = MsgBox(Date)
```

Uruchamiamy. Program w okienku pokazuje dzisiejszą datę w formacie zgodnym z systemowym np. yy-mm-dd (rok [dwie ostatnie cyfry]- miesiąc, dzień)

27.2 DateSerial

Zwraca w wewnętrznym formacie datę wpisaną przez nas ręcznie np.

```
Dim MyDate
MyDate = DateSerial(1969, 2, 12)
x = MsgBox (MyDate)
```

Zwraca to 69-02-12

27.3 DateAdd

Potrafi dodawać do podanej daty miesiące dni itd: Zobaczmy jak to działa. Do przycisku dodajmy kod:

```
Dim DataPoczatek As Date ' Zadeklarowanie zmiennych.
Dim TypDopisu As String
Dim Miesiace As Integer
Dim Odp
TypDopisu = "m"
DataPoczatek = InputBox("Podaj jakąś datę w formacie yy-mm-dd")
Miesiace = InputBox("Ile miesięcy chcesz dodać")
Odp = "Nowa data: " & DateAdd(TypDopisu, Miesiace, DataPoczatek)
x = MsgBox (Odp)
```

Uruchom go. Program najpierw pyta nas o datę, potem ile miesięcy chcemy dodać, i w końcu dostajemy odpowiedź.

Składnia tej funkcji:

```
DateAdd (Jednostka, Ilość jednostek, Data)
```

Yyyy	Rok
Q	Kwartał
M	Miesiąc

Y	Dni w roku
D	Dni
W	Weekendy
Ww	Tygodnie
H	Godziny
N	Minuty
S	Sekundy

Zauważ, że w naszym przykładzie użyliśmy literału m - czyli miesiące

27.4 DateDiff

Zwraca ilość jednostek czasu między dwoma datami

```
Dim TheDate As Date
TheDate = InputBox("Podzaj jakąś przyszłościową datę")
Odp = "Będzie to za: " & DateDiff("d", Now, TheDate) & " dni."
x = MsgBox (Odp)
```

Zauważ w tym programiku w linijce 4, że użyliśmy literki d co oznacza ilość dni.

```
Składnia :
DateDiff ("jednostka", data1, data2)
```

27.5 DatePart

Zwraca ilość jednostek licząc od początku okresu rozliczeniowego.

Okresem rozliczeniowym dla dni jest miesiąc, dla miesięcy rok, dla lat początek naszej ery. Funkcja zwraca po prostu części daty. Osobno dni, miesiące, lata, kwartały itp.

```
Dim TheDate As Date
Dim Odp
TheDate = InputBox("Podaj datę:")
Odp = "Od początku roki minęło : " & DatePart("q", TheDate) & " kwartałów"
x = MsgBox (Odp)
```

Składnia:

```
DatePart ("jednostka", data)
```

27.6 Now

Zwraca aktualną datę i czas

```
x = MsgBox (Now)
```

wyświetli to np. 00-06-15 11:12:34

27.7 Time

Zwraca aktualny czas np.

```
x = msgBox(Time)
```

zwróci to np. 12:23:54

27.8 TimeSerial

Zwraca w wewnętrznym formacie godzinę

```
MyTime = TimeSerial(16, 35, 17)' Reprezentacja godziny 4:35:17 po południu.
```

Zwróci to 16:35:17

27.9 Timer

Ta funkcja zwraca ona ilość sekund które upłynęły od północy.

Napiszemy program, który zapyta się nas ile to jest 12*6. Po podaniu prawidłowej odpowiedzi program wyświetli komunikat ile myśleliśmy nad tym zadaniem.

Na formularzu umieść przycisk i dodaj do niego kod (zwróć uwagę na pętlę Do Loop):

```
Dim CzasPrzed, CzasPo, CzasRoznica as Single
Dim  Odp as string
CzasPrzed = Timer  ' Pobieramy czas początku testu
Do
Odp = InputBox ("Ile wynosi iloczyn 12 i 6 ?")
Loop Until Val(Odp) = 72
CzasPo = Timer '' Pobieramy czas końca testu
CzasRoznica = CzasPo - CzasPrzed
x = MsgBox("Zajęło  ci to: " & CzasRoznica & " sekund")
```

przykład

27.10 Wyciąganie informacji o dacie i czasie

27.10.1 Funkcja Year (rok)

Zwraca wartość typu **Wariant** (**Liczba całkowita**), zawierającą liczbę całkowitą reprezentującą rok.

`Year(`*`data`*`)`

Wymagany argument ***data*** jest dowolną wartością typu Wariant, wyrażeniem liczbowym, wyrażeniem tekstowym lub dowolną ich kombinacją mogącą reprezentować datę. Jeśli argument ***data*** zawiera wartość Null, zostanie zwrócona wartość **Null**.

Przykład

W tym przykładzie funkcja **Year** służy do pobierania roku z określonej daty. W środowisku programistycznym literał daty jest wyświetlany zgodnie z formatem daty krótkiej, który jest zdefiniowany w ustawieniach regionalnych kodu.

```
Dim MojaData, MojRok
MojaData = #12 lutego 1969#     ' Przypisanie wartości daty.
MojRok = Year(MojaData)     ' Zmienna MojRok zawiera wartość 1969.
```

27.10.2 Funkcja Month (miesiąc)

Zwraca wartość typu **Wariant** (**Liczba całkowita**), określającą liczbę całkowitą z zakresu od 1 do 12, reprezentującą miesiąc roku.

Składnia

`Month(data)`

Przykład

W tym przykładzie funkcja **Month** służy do pobierania wartości miesiąca z określonej daty. W środowisku programistycznym literał daty jest wyświetlany w formacie daty krótkiej przy użyciu ustawień regionalnych kodu.

```
Dim MojaData, MojMiesiac
MojaData = #12 lutego 1969# ' Przypisanie wartości daty.
MojMiesiac = Month(MojaData) ' Zmienna MojMiesiac zawiera wartość 2.
```

27.10.3 Funkcja Day (dzień)

Zwraca wartość typu **Wariant** (**Liczba całkowita**), określającą liczbę całkowitą z zakresu od 1 do 31 włącznie, reprezentującą dzień miesiąca.

`Day(data)`

Wymagany argument ***data*** jest dowolną wartością typu Wariant, wyrażeniem liczbowym, wyrażeniem tekstowym lub dowolną ich kombinacją mogącą reprezentować datę. Jeśli argument ***data*** zawiera wartość Null, zostanie zwrócona wartość **Null**.

Przykład

W przykładzie pokazano zastosowanie funkcji **Day** do określenia dnia miesiąca dla wskazanej daty. W środowisku programistycznym literał daty jest wyświetlany zgodnie z formatem daty krótkiej, który jest zdefiniowany w ustawieniach regionalnych kodu.

```
Dim MojaData, MojDzien
MojaData = #12 lutego 1969# ' Przypisanie wartości daty.
MojDzien = Day(MojaData) ' Zmienna MojDzien zawiera wartość 12.
```

27.10.4 Funkcja Weekday (dzień tygodnia)

Zwraca wartość typu **Wariant** (**Liczba całkowita**), zawierającą liczbę całkowitą reprezentującą dzień tygodnia.

`Weekday(data [, pierwszy_dzień_tygodnia ] )`

Składnia funkcji **Weekday** zawiera następujące argumenty:

Argument	Opis
data	Wymagany. Wariant, wyrażenie liczbowe, wyrażenie tekstowe lub ich dowolna kombinacja, która może reprezentować datę. Jeśli argument data zawiera wartość Null, zostanie zwrócona wartość Null.
pierwszy_dzień_tygodnia	Opcjonalny. Stała określająca pierwszy dzień tygodnia. W przypadku pominięcia tego argumentu jest przyjmowana wartość vbSunday.

Ustawienia

Argument ***pierwszy_dzień_tygodnia*** ma następujące ustawienia:

Stała	Wartość	Opis
vbUseSystem	0	Użyj ustawienia API NLS.
vbSunday	1	Niedziela (domyślnie)
vbMonday	2	Poniedziałek
vbTuesday	3	Wtorek
vbWednesday	4	Środa
vbThursday	5	Czwartek
vbFriday	6	Piątek
vbSaturday	7	Sobota

Zwracane wartości

Funkcja **Weekday** może zwracać następujące wartości:

Stała	Wartość	Opis
vbSunday	1	Niedziela
vbMonday	2	Poniedziałek
vbTuesday	3	Wtorek
vbWednesday	4	Środa
vbThursday	5	Czwartek
vbFriday	6	Piątek
vbSaturday	7	Sobota

Przykład

W tym przykładzie funkcja **Weekday** została użyta do uzyskania dnia tygodnia na podstawie określonej daty.

```
Dim MojaData, MojDzienTygodnia
MojaData = #12 lutego 1969#    ' Przypisanie wartości daty.
MojDzienTygodnia = Weekday(MojaData)
' Zmienna MojDzienTygodnia zawiera wartość 4, ponieważ
' zmienna MojaData reprezentuje środę.
```

27.10.5 Funkcja WeekdayName

Zwraca **ciąg** wskazujący określony dzień tygodnia.

```
WeekdayName(dzień_tygodnia [, skrót ] [, pierwszy_dzień_tygodnia ] )
```

Składnia funkcji **WeekdayName** zawiera następujące argumenty:

Argument	Opis

dzień_tygodnia	Wymagany. Liczbowe oznaczenie dnia tygodnia. Wartości liczbowe, która odpowiadają poszczególnych dniom, zależą od ustawienia argumentu ***pierwszy_dzień_tygodnia***.
skrót	Opcjonalny. Wartość typu **Wartość logiczna**, która wskazuje, czy nazwa dnia tygodnia ma zostać skrócona. W przypadku pominięcia tego argumentu jest przyjmowana domyślna wartość **False**, co oznacza, że nazwa dnia tygodnia ma pozostać nieskrócona.
pierwszy_dzień_tygodnia	Opcjonalny. Wartość liczbowa, wskazująca pierwszy dzień tygodnia. Dostępne wartości są podane w sekcji Ustawienia.

Ustawienia

Argument ***pierwszy_dzień_tygodnia*** może mieć następujące wartości:

Stała	Wartość	Opis
vbUseSystem	0	Użyj ustawienia API obsługi języków narodowych (National Language Support, NLS).
vbSunday	1	Niedziela (domyślnie)
vbMonday	2	Poniedziałek
vbTuesday	3	Wtorek
vbWednesday	4	Środa
vbThursday	5	Czwartek
vbFriday	6	Piątek
vbSaturday	7	Sobota

27.10.6 Funkcja Hour (godzina)

Zwraca wartość typu **Variant** (**Integer**), określającą liczbę całkowitą z zakresu od 0 do 23 włącznie, reprezentującą godzinę.

```
Hour(godzina)
```

Wymagany argument ***godzina*** jest dowolną wartością typu Wariant, wyrażeniem liczbowym, wyrażeniem tekstowym lub dowolną ich kombinacją mogącą reprezentować godzinę. Jeśli argument ***godzina*** zawiera wartość Null, zostanie zwrócona wartość **Null**.

Przykład

W tym przykładzie za pomocą funkcji **Hour** jest pobierana wartość oznaczająca godzinę dla określonego czasu. W środowisku programistycznym literał czasu jest wyświetlany zgodnie z formatem godziny krótkiej, określonym w ustawieniach regionalnych kodu.

```
Dim MojCzas, MojaGodzina
MojCzas = #16:35:17# ' Przypisanie wartości czasu.
MojaGodzina = Hour(MojCzas) ' Zmienna MojaGodzina zawiera wartość 16.
```

27.10.7 Funkcja Minute

Zwraca wartość typu **Variant** (**Integer**) określającą liczbę całkowitą z zakresu od 0 do 59 włącznie, reprezentującą minutę.

`Minute(`*`godzina`*`)`

Wymagany argument ***godzina*** jest dowolną wartością typu Wariant, wyrażeniem liczbowym, wyrażeniem tekstowym lub dowolną ich kombinacją mogącą reprezentować godzinę. Jeśli argument ***godzina*** zawiera wartość Null, zostanie zwrócona wartość **Null**.

Przykład

W tym przykładzie użyto funkcji **Minute** w celu uzyskania wartości minut godziny z określonej wartości czasu. W środowisku programistycznym literał czasu jest wyświetlany w formacie czasu krótkiego, zgodnie z ustawieniami regionalnymi kodu.

```
Dim MojCzas, MojeMinuty
MojCzas = #16:35:17# ' Przypisanie wartości czasu.
MojeMinuty = Minute(MojCzas) ' Zmienna MojeMinuty zawiera wartość 35.
```

27.10.8 Funkcja Second

Zwraca wartość typu **Variant** (**Integer**), określającą liczbę całkowitą z zakresu od 0 do 59 włącznie, reprezentującą sekundę minuty.

`Second(`*`godzina`*`)`

Wymagany argument godzina jest dowolną wartością typu Wariant, wyrażeniem liczbowym, wyrażeniem tekstowym lub dowolną ich kombinacją mogącą reprezentować godzinę. Jeśli argument ***godzina*** zawiera wartość Null, zostanie zwrócona wartość **Null**.

Przykład

W tym przykładzie użyto funkcji **Second** w celu pobrania wartości sekund minuty z określonej wartości czasu. W środowisku programistycznym literał czasu jest wyświetlany w formacie czasu krótkiego, zgodnie z ustawieniami regionalnymi kodu.

```
Dim MojCzas, MojeSekundy
MojCzas = #16:35:17# ' Przypisanie wartości czasu.
MojeSekundy = Second(MojCzas) ' Zmienna MojeSekundy zawiera wartość 17.
```

27.11 Zadania do działu operacje na dacie i czasie

Zadanie 1

Wykonaj program, który będzie podawał czas pomiędzy dwoma kliknięciami w obiekty na userformie (np. przyciski Command Buton). Niech program zapamiętuje najkrótszy czas pomiędzy dwoma kliknięciami. Umożliwi to rywalizację o najkrótszy czas. Zwróć uwagę, że muszą to być oddzielne obiekty, ponieważ klikanie w pojedynczy obiekt zostanie zarejestrowane jako double click.

Rozwiązanie

Zadanie 2

Wykonaj program, który będzie zadawał zadania obliczeniowe. Niech będzie to pięć lub sześć pytań. Program ma ponawiać pytania aż do podania przez użytkownika prawidłowej odpowiedzi. Niech po odpowiedzi na wszystkie pytania program napisze ile czasu zajęła nam odpowiedź na wszystkie pytania razem oraz ile czasu potrzebowaliśmy na odpowiedź na poszczególne pytania. Niech program wystawi ocenę uzależnioną od czasu, jaki zajęła nam odpowiedź na pytania.

W zadaniu wykorzystaj tabelę danych.

Rozwiązanie – dokładnie taki sam problem, ale pytania z historii, gdzie trzeba podać daty. Pytania o wyniki działań arytmetycznych są w tym wypadku logiczniejsze, bo jest większa szansa na dojście do prawidłowego wyniku.

Zadanie 3

Wykonaj program podający czas do jakiejś daty wprowadzonej do programu. Jeśli potrafisz to niech data, do której będzie odmierzany czas, będą wartościami podawanymi przez użytkownika.

Rozwiązanie - wersja z kontrolą wprowadzonych liczb i spinbuttonami. Bez kontroli pełnych lat i miesięcy. Jak widać funkcja DateDiff podaje różnicę tylko w wybranej jednostce czasowej.

Rozwiązanie – wersja rozbudowana, która potrafi sprawdzić upływ pełnych, lat, miesięcy itd.

Zadanie 4

Wykonaj program zliczający czas kolejnych dziesięciu kliknięć w dowolny obiekt. Niech po dziesiątym kliknięciu pojawi się czas. Gromadź czasy na tablicy wyników.

Rozwiązanie

Zadanie 5

Wykonaj grę **BDZIĄK** jak na przykładzie bądź podobną, w której będziesz rejestrować upływający czas a sukces bądź porażka będą zależały od spełnienia warunku czasowego.

Rozwiązanie/przykład

28 Funkcje matematyczne

Podstawowe funkcje matematyczne.

1. Abs (liczba) - zwraca wartość bezwzględną z liczby np:

```
Abs(4) = 4
Abs(-4) = 4
Abs(0) = 0
```

2.Atn (liczba) - zwraca wartość arcusCotangens liczby wyrażoną w radianach np.

4 * Atn(1) = pi

3.Cos (liczba) - zwraca cosinus z liczby

4.Exp(liczba) - e ^ liczba ; e - podstawa logarytmu naturalnego

5.Log (liczba) - zwraca wartość logarytmu naturalnego z liczby

6.Sng(liczba) - Dla liczby dodatniej zwraca 1, dla ujemnej -1, a dla 0 - 0 np.

```
Sng(23) = 1
Sng(-29) = -1
Sng(0) = 0
```

7.Sin(liczba) - zwraca sinus z liczby

8.Sqr(liczba) - zwraca pierwiastek kwadratowy z liczby np.

```
Sqr(9) = 3
```

Jeśli chcesz wyliczać pierwiastki n-tego stopnia z liczby x to musisz zastosować wzór: x ^ (1/n)

9.Tan(liczba) - zwraca tangens liczby

10. INT((6 * RND) + 1) - całkowita liczba losowa z przedziału od 1 do 6. No może nie całkiem losowa - sam zobacz. Uruchom VB, na formie ułóż przycisk i dodaj do niego następujący kod:

```
Dim x
x = INT((6 * RND) + 1)
odp = MsgBox(x)
```

Wystartuj program. Po naciśnięciu przycisku pojawi się pierwsza liczba - zapamiętaj ją. Teraz OK i jeszcze raz kliknij na przycisk - tym razem inna (chyba) liczba, także ją zapamiętaj. Zrób tak jeszcze kilka razy (zapamiętaj tylko te dwie pierwsze). No i co ? Są liczby losowe. Teraz wyłącz program i włącz go jeszcze raz. Kliknij na przycisk - pierwsza liczba jest taka sama jak poprzednio. Druga też... Jakbyś sprawdzał dalej to wszystkie by się powtórzyły w tej samej kolejności. Aby VB za każdym razem losował inne liczby należy przed funkcją RND dopisać Randomize, będzie to wyglądało tak:

```
Dim x
Randomize
x = INT((6 * RND) + 1)
odp = MsgBox(x)
```

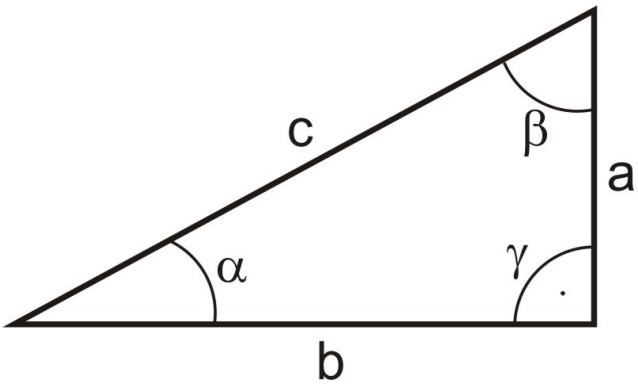
Inne funkcje:

```
Secant Sec(X) = 1 / Cos(X)
Cosecant Cosec(X) = 1 / Sin(X)
```

```
Cotangent Cotan(X) = 1 / Tan(X)
Inverse Sine Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Inverse Cosine Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Inverse Secant Arcsec(X) = Atn(X / Sqr(X * X – 1)) + Sgn((X) – 1) * (2 *
Atn(1))
Inverse Cosecant Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) – 1) * (2
* Atn(1))
Inverse Cotangent Arccotan(X) = Atn(X) + 2 * Atn(1)
Hyperbolic Sine HSin(X) = (Exp(X) – Exp(-X)) / 2
Hyperbolic Cosine HCos(X) = (Exp(X) + Exp(-X)) / 2
Hyperbolic Tangent HTan(X) = (Exp(X) – Exp(-X)) / (Exp(X) + Exp(-X))
Hyperbolic Secant HSec(X) = 2 / (Exp(X) + Exp(-X))
Hyperbolic Cosecant HCosec(X) = 2 / (Exp(X) – Exp(-X))
Hyperbolic Cotangent HCotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) – Exp(-X))
Inverse Hyperbolic Sine HArcsin(X) = Log(X + Sqr(X * X + 1))
Inverse Hyperbolic Cosine HArccos(X) = Log(X + Sqr(X * X – 1))
Inverse Hyperbolic Tangent HArctan(X) = Log((1 + X) / (1 – X)) / 2
Inverse Hyperbolic Secant HArcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Inverse Hyperbolic Cosecant HArccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) +
1) / X)
Inverse Hyperbolic Cotangent HArccotan(X) = Log((X + 1) / (X – 1)) / 2
Logarithm to base N LogN(X) = Log(X) / Log(N)
```

Zadanie:

Wykonaj program, który będzie potrafił wyliczyć długość boku **b** w trójkącie prostokątnym jak na rysunku. Długość **C** oraz kąt **alfa** ma wpisać użytkownik programu.



Rozwiązanie

29 Funkcje łańcuchowe, czyli operacje na ciągach znaków.

Funkcje łańcuchowe są to funkcje które wykonują operacje na literach i znakach.

29.1 Chr – użycie znaku w kodzie ASCII.

Chr (liczba) - zwraca znak ASCII odpowiadający argumentowi liczba.

Wyjaśnienie! Jak wiemy komputer „pamięta" wszystko w postaci liczb binarnych. Aby używać tekstu stworzono systemy kodowania znaków. System kodowania znaków to nic innego jak przypisanie każdemu znakowi konkretnej liczby. Do tej pory największe znaczenie miał system ASCII, w przyszłości obowiązującym standardem stanie się UNICODE. W systemie ASCII korzystamy z liczb 8 bitowych. Za pomocą 8 bitów możemy zapisać liczbę od 0 do 255 zatem mamy do dyspozycji do 256 różnych znaków. Tyle wiadomości na ten moment wystarczy.

Napiszmy mały programik, który uzmysłowi kiedy będzie potrzebne polecenie CHR:

Uruchom VB - na formie ułóż przycisk. Zakładam, że chcemy wyświetlić następujący komunikat:

```
Powieść „Lalka" to jedna z najwybitniejszych powieści Bolesława Prusa.
```

Jak wiemy znak cudzysłowu służy do poinformowania programu, że za nim znajdują się znaki do wyświetlenia, bez ich analizowania, a następny cudzysłów kończy ten ciąg znaków. Zatem jeśli chcemy wyświetlić w komunikacie programu znak cudzysłowu to musimy uczynić to sposobem. Wpisanie polecenia:

```
x = msgbox ("Powieść „Lalka" to jedna z najwybitniejszych powieści
Bolesława Prusa." )
```

nie zadziała, program będzie próbował interpretować fragment …***Lalka***… i pojawi się błąd.

Wewnątrz wyświetlanego komunikatu musimy użyć zatem polecenia CHR i za jego pomocą wyświetlić znak cudzysłowu. Znak cudzysłowu ma numer 34. Poprawnie wyświetlić ten komunikat będzie można poleceniem:

```
x = msgbox ( " Powieść " & chr(34) & "Lalka" & chr(34) & " to jedna z
najwybitniejszych powieści Bolesława Prusa." )
```

Trochę bardziej skomplikowane - ale tak musi być

Inny przykład. Chcemy „złamać wiersz" tak aby tekst, np. w Input Box, pokazywał się w dwóch wierszach. Wstawiamy znak nr 10, czyli **New Line** (nowa linia).

Przykład:

```
x = InputBox("Podaj datę " & Chr(10) & " odsieczy wiedeńskiej (Bitwy pod
Wiedniem)?", "Pytanie")
```

29.2 Zmiana WIELKIE/małe litery

LCase (string) - zmienia wszystkie znaki w stringu na małe np:

```
x = "Stonoga Ma 100 nóg"
y = LCase (x)
```

Da w wyniku "stonoga ma 100 nóg"

UCase - zmienia wszystkie znaki w stringu na duże np.

```
x = "Stonoga Ma 100 nóg"
y = UCase (x)
```

Da w wyniku "STONOGA MA 100 NÓG"

29.3 Wycinanie fragmentu Stringa.

Left (string, n) - zwraca n pierwszych znaków łańcucha (stringa) np.

```
x = "Stonoga Ma 100 nóg"
y = Left (x, 10)
```

Zwróci to - Stonoga Ma

Right (string, n) - zwraca n ostatnich znaków łańcucha np.

```
x = "Stonoga Ma 100 nóg"
y = Right (x, 7)
```

Zwróci - 100 nóg

Mid (string, n,m) - zwróci będący częścią argumentu łańcuch począwszy od znaku numer n do znaku nr. m (licząc od n !) np.

```
x = "Stonoga Ma 100 nóg"
y = Mid (x, 9,6)
```

Zwróci - 100 nóg

S	t	O	n	o	g	a		M	A		1	0	0		n	ó	G
1	2	3	4	5	6	7	8	**9**	**1**	2	3	4	5	**6**			

Zauważ, że litera M jest znakiem nr 9 ciągu znaków, z którego korzystamy jak i pierwszym ciągu który chcemy wyciąć!

29.4 Obliczenie długości stringu.

Len(string) - zwraca długość stringu np.

```
x= "Stonoga"
y = Len(x)
```

W tym przypadku y = 7

29.5 Usuwanie spacji

(L),(R)Trim - usuwa spacje wiodące - L z lewej, R z prawej, samo Trim i z lewej i z prawej np

```
x = Ltrim ("       to jest lekcja VB      ")
```

zwróci "to jest lekcja VB   "

```
x = Rtrim ("       to jest lekcja VB      ")
```

zwróci "   to jest lekcja VB"

```
x = trim ("       to jest lekcja VB      ")
```

zwróci "to jest lekcja VB"

29.6 Zamiana znaku na liczbę ASCII (ASC).

ASC (znakASCII) - odwrotność funkcji Chr - czyli ze znaku robi liczbę np

```
y = Asc("A")
zwróci liczbę 65
```

Zadanie1:

Napisz program, który wyświetli wszystkie znaki ASCII w ListBox'ie włącznie z podaniem ich numerów.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Tabela 1Znaki ASCII podstawowe

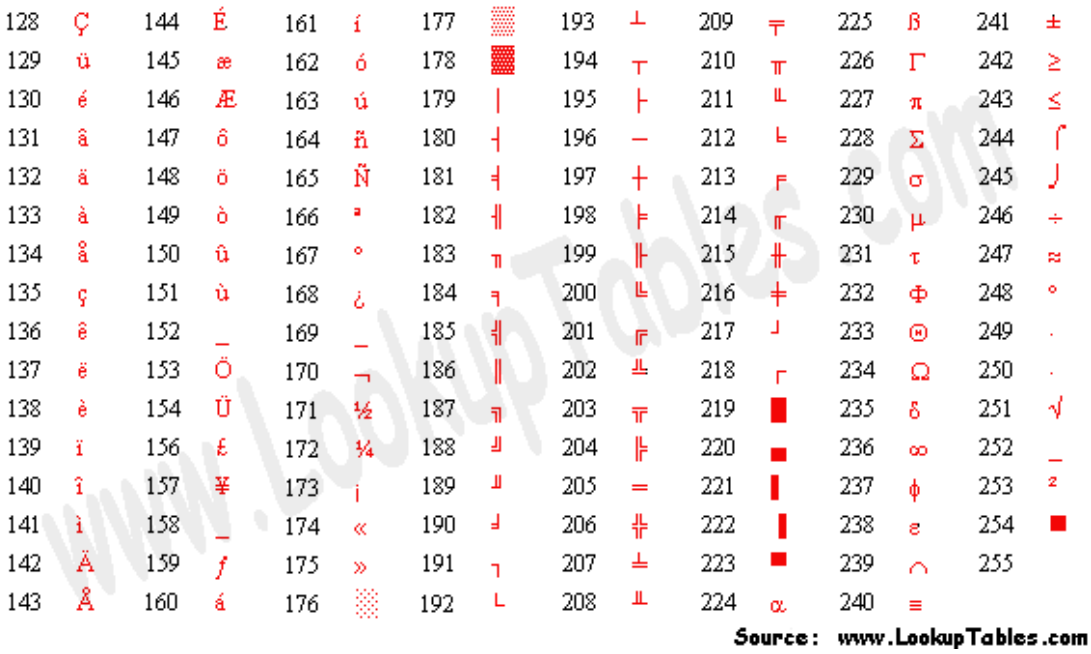


Tabela 2 Znaki ASCII dodatkowe - zależne od włączonej strony kodowej.

Zadanie 2:

Napisz program, który będzie gromadził wpisywane przez Ciebie wyrazy w ComboBox'ie. Jednocześnie ma przepisywać wyrazy napisane z wielkiej litery do sąsiedniego ListBox'a.

Oprócz tego program powinien zliczać wszystkie wyrazy wpisane do ListBox'a oraz wszystkie wpisane do ComboBox'a.

Rozwiązania

Sprawdzian: Czas, matematyka i stringi.

30 Testowanie i konwersja danych

30.1 Funkcje testujące dane

Funkcje testujące dane nadają się doskonale do obsługi błędów w programie.

Przykład:

Napiszmy program, który będzie podnosił do potęgi liczbę wpisaną w okienko tekstowe.

Wstaw na UserFormie textbox-a nazwij go txtLiczba. Teraz jeszcze przycisk - jako etykietę nadaj mu "Podnieś do kwadratu". OK. Kliknięcie na przycisku ma obliczyć kwadrat liczby wpisanej w TextBox i podać wynik w MsgBox-ie. Do przycisku dodaj kod:

```
Dim x
x = Val(txtLiczba) ^ 2
odp = MsgBox(txtliczba & " do kwadratu to " & x)
```

Uruchamiamy program - w okienko tekstowe wpisujemy jakąś liczbę i działa. Jeśli jednak w okienko tekstowe wpiszemy jakieś słowo np. Visual Basic. Wyskoczy nam komunikat, że Visual Basic do kwadratu to 0. Nonsens! Oczywiście moglibyśmy skorzystać z pętli i wymusić wpisanie poprawnej danej. Poznamy jednak inne rozwiązanie.

Jeśli użytkownik wpisze w pole tekstowe stringa zamiast liczby to program wyświetli komunikat, że jest błąd. Do rozpoznania czy w polu znajduje się string czy liczba posłużą funkcje testujące dane. Nasz program teraz powinien wyglądać tak:

```
Dim x
If IsNumeric(txtliczba) Then
x = Val(txtliczba) ^ 2
odp = MsgBox(txtliczba & " do kwadratu to " & x)
Else
odp = MsgBox("Błąd", vbCritical, "Błąd")
End If
```

Przykład

Co można rozpisać jako:

```
Zapamiętuj x
Jeśli to co w polu tekstowym to liczba to...
Obliczenie + Komunikat z odpowiedzią
W przeciwnym przypadku
Błąd
```

Najważniejsze funkcje testujące dane:

1. IsDate (string) - zwraca true jeśli data
2. IsNumeric (string) - zwraca true jeśli liczba
3. IsEmpty (zmienna) - zwraca True jeśli zmienna nie istnieje

Przykład użycia IsEmpty

```
Dim zmienna
If IsEmpty (zmienna) Then
MsgBox ("Błąd")
End If
```

W tym przypadku wyskoczy komunikat Błąd ponieważ nigdzie nie mamy podstawienia czegoś pod zmienną.

30.2 Funkcje konwersji danych

Stosuje się je dla danych tak aby zmienić ich wartość albo umożliwić wykonanie na nich operacji możliwych tylko dla określonych typów danych.

Przykład:

```
x = 1/7   'daje to w wyniku 0,1428571
x = CDbl (1/7)  'daje w wyniku 0,142857142857143
```

VB domyślnie stosuje tryb Single - pierwszy przypadek. Po konwersji danych na Double otrzymujemy dokładniejszy wynik...

Podstawowe funkcje konwersji danych:

Cint (x) - Zaokrągla argument z częścią ułamkową do liczby całkowitej wg zasady od x>=0 do x<=0,5 zaokrąglenie do zera, czyli CInt(x)=0, x>0,5 do x<=1 zaokrąglenie do jednego, czyli CInt(x)=1. przykłady: CInt (12,7) = 13, CInt(10,5) = 10, Cint(10,50001)=11, Cint(4,2)=4.

Int(x) - Zaokrągla w dół argument z częścią ułamkową do liczby całkowitej np. int (12,7) = 12

CStr (x) - Konwertuje argument na stringa

CVar (x) - Konwertuje argument na Variant

Fix (liczba) - Obcina część ułamkową

Hex(liczba) - Konwertuje numeryczny argument na heksadecymalny

Oct (liczba) - Konwertuje numeryczny argument na oktalny

Zwróć uwagę, że Fix i Int nie działają tak samo. Dla sytuacji, kiedy zamieniamy liczby dodatnie: Fix (12,5) = 12 i Int(12,5) = 12

Fix = Int

Jeśli jednak liczba jest ujemna:

Fix (- 12,5) = -12 ' Bo obcina to co jest po przecinku

Int (-12,5) = -13 'Bo -12 było by większe niż -12,5, a ma być mniejsze więc wynikiem jest -13

31 Polecenie Shell

Polecenie Shell umożliwia uruchamianie innych programów z VB,

Jeśli chcemy np. uruchomić Painta z poziomu programu napisanego przez nas musimy użyć funkcji shell.

Przykład:
Uruchom VB, ułóż przycisk i dodaj do niego kod:

```
x = shell("C:\Windows\system32\Mspaint.exe,1)
```

Składnia funkcji Shell:

```
x = Shell(Ścieżka dostępu, styl okna)
```

Ścieżka dostępu to adres pliku wykonywalnego, który chcemy uruchomić. Dostępne style okna podano w poniższej tabelce.

0	Okno uruchamianego programu zostaje ukryte (trochę bez sensu)
1	Normalne okno
2	Okno zminimalizowane
3	Okno zmaksymalizowane
4	Normalne okno niezaznaczone
6	Zminimalizowane okno nie zaznaczone

Tabela 3 Style okna

To nie błąd - w tabelce nie ma wartości 5.

Przykład

32 Sterowanie listami napędów, katalogów, plików (nie działa w VB for Apps)

Z listami mamy często do czynienia w systemie operacyjnym? Otwórz eksplorator windows - Start > Programy > Eksplorator Windows

Po lewej stronie widzisz drzewo katalogów - właśnie to jest lista katalogów, Lista plików jest po prawej stronie, a lista dysków powyżej. To co oferuje Eksplorator jest trochę doskonalsze od list w VB.

Zasada działania. Mamy listę dysków i katalogów. Wybór dysku na liście dysków ma wyświetlić katalogi wybranego dysku a wybór katalogu ma spowodować pokazanie plików w nim zawartych.

Na formie układamy - Listę dysków (DriveListBox - ikonka znajduje się obok ikonki zegarka - To ten szary prostokącik z kreską i kropką. Następne listę katalogów, (ikonka - żółty katalog), oraz listę plików, to zaraz obok ikonki katalogu.

W liście napędów (domyślnie wpisane jest tam C oraz nazwa dysku) dodaj kod:

```
Dir1 = Drive1
```

A w liście katalogów:

```
File1 = Dir1
```

Uruchom program. Teraz powinno wszystko działać.

Program zwraca ścieżkę do pliku. Aby to sprawdzić ułóż na formie jeszcze Etykietę (Label) i nazwij ją lblWynik. Do listy katalogów dopisz jeszcze (pod File1 = Dir1):

```
lblWynik = Dir1.Path
```

Spowoduje to wyświetlanie się aktualnej ścieżki do pliku. Aby jeszcze wyciągnąć nazwę wybranego pliku w liście plików dopisz:

```
lblWynik = Dir1.Path & "\" & File1.FileName
```

33 Tworzenie i modyfikacje plików.

Uwaga! Zalecam tworzenie programów wykorzystujących poniższe opcje z użyciem zewnętrznego nośnika danych np. Dyskietka 3,5” (Litera dysku A:\)

Tworzenie plików i zapisywaniem do nich informacji, czyli sposób w jaki można gromadzić wyniki działania własnego programu.

33.1 Open - otwieranie plików

Plik można otworzyć funkcją Open. Oczywiście jest to konieczne, aby móc edytować plik musimy go najpierw otworzyć.

Składnia:

```
Open "C:\katalog\plik.roz" for Mode as # number
```

Mode określa to co będziemy chcieli zrobić z plikiem:

Input	Jedynie odczyt danych (najbezpieczniejsza funkcja bo nic nie zepsujemy)
Output	Dopisywanie do pliku informacji
Append	Nadpisywanie informacji

Number to zaś liczba porządkowa otwartego pliku. Nie możemy otworzyć dwóch plików z tym samym numerem porządkowym! Czyli:

```
Open "C:\moje dokumenty\index.dat" for input as #1
Open "C:\info.txt" for input as #1
```

Coś takiego jest niemożliwe! Można tak:

```
Open "C:\moje dokumenty\index.dat" for input as #1
Open "C:\info.txt" for input as #2
```

33.2 Close – zamykanie plików

Funkcja Close zamyka pliki. Czyli jeśli użyjemy funkcji Open to także musimy użyć funkcji close. Jeśli chcemy zamknąć plik z numerem porządkowym 3 to piszemy:

```
Close 3
```

Jeśli chcemy zaś zamknąć wszystkie aktualne pliki to po prostu wpisujemy close. Czy więc teraz jest możliwa taka sytuacja?

```
Open "C:\moje dokumenty\index.dat" for input as #1
Close 1
Open "C:\info.txt" for input as #1
```

TAK ! Otwieramy plik z #1 - zamykamy go (teraz #1 się zwalnia), no i znowu możemy otworzyć plik z #1

No dobra, ale po co tak otwierać i zamykać pliki ? Po to aby można było z nich skorzystać !!!

33.3 Write – zapisywanie do plików

Instrukcja ta zapisuje do pliku pewne informacje. Jak ? Napiszemy teraz program zapisujący do pliku imiona pewnych dzieci. Otwieramy więc VB - tworzymy przycisk i dodajemy kod:

```
Dim x
Dim dzieci(6)    'Deklaracja tablicy
dzieci(1) = "Kasia"
dzieci(2) = "Michał"
dzieci(3) = "Piotruś"   'Nie musimy wykorzystywać wszystkich 6 imion
Open "C:\dzieci.txt" for Output as #1   'Output bo będziemy zapisywać do
pliku
For x = 1 to 3  'Bo mamy troje dzieci
Write #1, dzieci(x)
Next x
Close 1
```

Objaśnienie:

```
Dim-y  -  to chyba wiadomo, pierwszy (x) to zmienna dla For-a, a drugi to
tablica
Następnie przypisanie dzieci do tablicy
Open - tutaj zostaje otwarty plik do zapisu z liczbą porządkową #1
For - Od x = 1 do 3
Zapisz do #1 zmienną dziecko(x); teraz x = 1 (patrz FOR) a więc Kasia
Next - powrót do for ale x = 2
For ...
Zapisz do #2 zmienną dziecko(x); teraz x = 2 a więc Michał...
Tak jeszcze przez Piotrusia i
Zamknij plik #1
```

Teraz wystarczy sprawdzić czy na dysku C znajduje się plik dzieci.txt. Otwórz go klikając na niego. W pliku powinny się znaleźć 3 imiona w cudzysłowach? W VB stringi podaje się w cudzysłowach, liczby bez i taki też sposób zapisu został zastosowany w wygenerowanym pliku.

Gdybyśmy zapomnieli użyć funkcji Close plik nie zostałby zapisany - a więc zostałby utworzony (funkcja Open as Output), ale nie zapisany - byłby więc pusty.

Problem: Póki wiemy ile jest informacji do zapisu sprawa jest prosta (For x = 1 to ilość). A jeśli ilości nie znamy, albo będzie się stale zmieniała musimy zastosować coś takiego:

```
Dim x, wynik as single
Dim tablica(100)
... - tutaj np. zapisy do tablicy
Open "C:\plik.roz" for Mode as # number
For x = 1 to 100000  'wpisujemy jakąś liczbę większą od rozmiaru tablicy
wynik = VarType(tablica(x))
If wynik = 0 then Exit For
Write # number, tablica(x)
Next x
Close 1
```

Objaśnienie: Pętla For - Next przerwie się jeśli wynik = 0 (IF wynik = 0 ...). Funkcja VarType zwróci zero (na temat tej funkcji więcej wiadomości na razie nie jest potrzebne) co zostanie zinterpretowane jako błąd i wyjdziemy z pętli. Koniec zapisu.

Funkcją write można zapisać kilka informacji w jednej linijce np.

```
Write #1, string, jakasliczba, jakasdata
```

Spowoduje to zapisanie do pliku informacji w formie jak pokazano poniżej:

```
" To jest jakiś string", 123456789876, #12-11-99#
```

33.4 Input - Odczytywanie informacji z plików.

Odczytamy trzy imiona z pliku dzieci.txt. Na formie ułóżmy przycisk i dodajmy do niego kod:

```
Dim x
Dim dzieci(10)
Open "C:\dzieci.txt" for input as #1
For x = 1 to 3
Input #1, dzieci(x)
odp = MsgBox(dzieci(x))
Next x
Close
```

Spowoduje to trzykrotne wyświetlenie się MsgBox-a z imieniem kolejnego dziecka. Funkcja Input ma identyczną składnię jak Write. Przeanalizuj kod. Jeśli masz z tym problem wróć do poprzednich tematów.

W sytuacji jeśli chcemy odczytać wszystkie rekordy z pliku, ale nie wiemy ile ich jest musimy użyć funkcji EOF (**E**nd **O**f **F**ile). Działa ona na zasadzie sprawdzenia czy jest jeszcze coś w pliku?

Program ma wyglądać tak:

```
Czy jest jeszcze coś w pliku ? Jeśli tak to...
Odczytaj i powrót
Jeśli nie to koniec

Składnia funkcji EOF
EOF(numer pliku)
```

Należy tu użyć pętli DO-LOOP. Kod powinien wyglądać tak:

```
Dim tablica(100)
Open "c:\dzieci.txt" for input as #1
Do Until EOF(1) = True
Input #1, tablica(x)
Loop
Close
```

Objaśnienie:

```
Zapisz tablicę 100 - elementową
Otwórz plik dzieci.txt do odczytu danych
Dopóki EOF nie równe True czyli równe False !!! (czyli dopóki nie koniec
pliku)
Odczytuj i zapisuj do tablicy
Kontynuuj
Zamknij plik
```

Jeśli chcemy odczytać wybrane linijki np. drugą i trzecią linijkę pliku dzieci.txt musimy zastosować następującą procedurę:

```
Dim dziecko2, dziecko3 as string
Open "c:\dzieci.txt" for input as #1
Input #1, dziecko2
Input #1, dziecko2
Input #1,dziecko3
Close
```

Działa to tak:

Zapamiętuj dziecko2,dziecko3

Otwórz plik do odczytu danych

Zapisz do zmiennej dziecko2 pierwszą linijkę pliku #1. Nam chodzi o drugą.

No to jeszcze raz zapisz do zmiennej dziecko2 drugą linijkę pliku #1

I do zmiennej dziecko3 trzecią linijkę pliku.

Przykład

Zadanie: Utwórz program, który będzie umożliwiał wpisywanie do ComboBox'a różnych wyrażeń tekstowych (maksymalnie 100 wyrażeń). Wstaw przycisk, którego kliknięcie będzie uruchamiało procedurę zapisującą wyrażenia do pliku tekstowego o nazwie „lista.txt".

Wstaw drugi przycisk po to aby umożliwić odczytanie wyrażeń pliku tekstowego „lista.txt" i wyświetlenie ich w polu ListBox.

Rozwiązanie

33.5 Kill - Usuwanie plików i katalogów

Usuwanie plików. Służy do tego funkcja kill. Używamy jej w następujący sposób:

```
Kill "C:\dzieci.txt"   ' Usunie ona plik dzieci.txt zlokalizowany na dysku
c:
Kill "C:\Bzdety\*.*  'Usunie wszystkie pliki z katalogu Bzdety
Kill "C:\bzdety\*.txt"   ' Usunie wszystkie dokumenty tekstowe z katalogu
bzdety
Kill "C:\bzdety\A*.*  'Usunie wszystkie pliki z katalogu bzdety zaczynające
się na literę a
```

Usuwanie katalogów. Służy do tego funkcja RmDir - usuwa ona katalogi (tylko puste).

RmDir "C:\bzdety" 'Spowoduje usunięcie katalogu bzdety jeśli nie ma w nim żadnych plików

Jeśli chcemy usunąć katalog z plikami? Stosujemy połączenie obu wcześniej poznanych komend:

```
Kill "C:\bzdety\*.*"
RmDir "C:\bzdety"
```

Niestety, jeśli w katalogu bzdety będą jakieś podkatalogi (podfoldery), to wyskoczy błąd.

33.6 MkDir – tworzenie katalogów

Do tworzenia katalogów służy funkcja MkDir np:

MkDir "C:\Moje dokumenty\klasa" 'Spowoduje utworzenie katalogu klasa w Moich dokumentach.

33.7 Funkcja Name

Funkcja Name służy do: zmieniania nazw plików, katalogów i przenoszenia ich w inne miejsce na dysku.

Aby zmienić nazwę pliku:

```
Name "C:\Moje dokumenty\Do mamy.doc" as "C:\Moje dokumenty\Wysłany do
mamy.doc"
```

Aby zmienić nazwę katalogu:

```
Name "C:\Bzdety" as "C:\głupoty"
```

Aby przenieść plik w inne miejsce (od razu zmieniając jego nazwę)

```
Name "C:\dzieci.txt" as "C:\Moje dokumenty\ludzie.txt"
```

Można oczywiście łączyć ciągi znaków ze zmiennymi aby móc na bieżąco definiować nazwy.

Przykład:

```
Dim nazwa, nowanazwa
nazwa = TextBox1.Text
nowanazwa = TextBox2.Text
Name "C:\" & nazwa As "C:\" & nowanazwa
```

Objaśnienie:

Deklaracja zmiennych

Program pobiera nazwę wpisaną do TextBoxa1 i zapisuje do zmiennej nazwa

Program pobiera nazwę wpisaną do TextBoxa2 i zapisuje do zmiennej nowanazwa

Program zmienia nazwę pliku na dysku C:\nazwa na C:\nowanazwa, czyli zmieni nazwę pliku z takiej jaka jest wpisana w TextBox1 na nazwę wpisaną w TextBox2.

Zadanie: Utwórz program, który stworzy katalog C:\zadanie i w tym katalogu umieści 100 plików o nazwach kolejno plik1.txt, plik2.txt, plik3.txt… i tak dalej. W każdym z plików niech znajdzie się jeden wpis. Niech będzie to numer pliku.

Rozwiązanie

Zadanie: Utwórz program, który automatycznie po uruchomieniu zacznie tworzyć na dysku C: katalogi o nazwie zadanie i kolejnych numerach (zadanie1, zadanie2 itd.). W każdym katalogu umieści 1000 plików o nazwach kolejno plik1.txt, plik2.txt, plik3.txt… i tak dalej. W każdym z plików niech znajdzie się jeden wpis. Niech będzie to numer pliku.

34 Dodatkowe ćwiczenia z wykorzystaniem VB

34.1 Generowanie liczb losowych

Generowanie liczb losowych może pomóc w realizacji wielu zadań. Np. testowanie poprawności działania algorytmów, symulowanie wydarzeń rzeczywistych i inne. Zobacz program losujący ucznia do odpowiedzi.

Looser ucznia exe

Looser ucznia. Losowanie ucznia inna wersja bezpośrednio w excelu

34.2 Szyfrowanie

Po przeprowadzeniu lekcji z Systemów operacyjnych z zakresu szyfrowania wiadomości można napisać program realizujący szyfrowanie.

Zadanie 1 Napisz program szyfrujący tekst szyfrem cezara. Rozwiń potem program do postaci, w której będzie można zmieniać przesunięcie liter. Dla uproszczenia przyjmujemy, że używamy tylko małych liter i bez polskich znaków diakrytycznych. Znaków spacji nie kodujemy tylko zostawiamy bez zmian.

Podpowiedź: Musisz skorzystać z działu poświęconemu ciągom tekstowym.

szyfrowanie - szyfr cezara

Zadanie 2 Napisz program szyfrujący szyfrem Vigenere'a. Słowo kluczowe musi być możliwe do wpisania w programie. Upraszczamy kodowanie ograniczając się do małych liter i nie używamy polskich znaków. Znaki spacji pozostawiamy bez kodowania.

szyfrowanie – szyfr Vigenere'a (bardziej złożony program, przerabiać jako drugi)

Założenia:

1. używamy zestawu znaków kodu ASCII od 97 - 122 (tylko małe litery)

2. słowo kluczowe definiuje przesunięcie zgodnie ze wzorem a=1, b=2 itd. UWAGA! inaczej niż w załączonej tablicy, gdzie a = przesunięcie o 26

3. nie używamy polskich znaków (co wynika już z punktu 1).

4. spację pozostawiamy bez zmian (tzn spacji nie kodujemy)

5. nawet jeśli nie kodujemy spacji to zużywamy na nią literę z klucza.

np. tekst: **nasza klasa**

klucz: **as**

przyporządkowanie liter klucza do tekstu

1	2	3	4	5	6	7	8	9	10	11
n	a	s	z	a		k	l	a	s	a
a	s	a	s	a	s	a	s	a	s	a

Tablica Vigenere'a

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
26	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

35 Sprawdziany

35.1 Sprawdzian z podstaw

Sprawdzian z podstaw do przeprowadzenia po lekcji z Rozszerzenie funkcji InputBox.

Sprawdzian z podstaw - poprawka

Zadania są bardzo proste więc nie potrzeba przygotowywać rozwiązań przykładowych.

35.2 Sprawdzian – zadania obliczeniowe.

Po lekcji: Zmiana właściwości obiektów (komponentów) kodem.

Sprawdzian - zadania obliczeniowe

Zadania od 1 do 10 na pierwszy termin.

Zadania A-E na poprawę sprawdzianu.

Rozwiązania – zadania obliczeniowe

Omówienie – zadanie 1

35.3 Sprawdzian z zakresu listbox, call, global, frame (option button), tablice.

Po lekcji Tablice.

Sprawdzian z zakresu listbox, call, global, frame (option button)

Rozwiązania – niekompletne ponieważ zadania są banalne.

To jest sprawdzian na koniec semestru pierwszego.

Poprawa sprawdzianu ListBox

Rozwiązania sprawdzian ListBox poprawa

35.4 Sprawdzian teoretyczny na koniec pierwszego semestru

Po przerobieniu działu tablice.

Sprawdzian test + pytania otwarte (2 grupy po 12 pytań – 13 punktów)

Sprawdzian poprawkowy (12 pytań - 12 punktów)

Sprawdzian poprawkowy 2 (10 pytań – 10 punktów)

35.5 Sprawdzian: Czas, matematyka i stringi.

Po rozdziale: Zamiana znaku na liczbę ASCII (ASC).

Zadania na sprawdzian

Rozwiązania

Sprawdzian poprawkowy

Rozwiązania zadań ze sprawdzianu poprawkowego

35.6 Końcowy sprawdzian praktyczny

sprawdzian (na pierwszy termin zadania 1-6)

rozwiązania zadań ze sprawdzianu (od 1 do 6)

rozwiązanie zadania 7

rozwiązanie zadania 8

sprawdzian końcowy praktyczny poprawkowy

sprawdzian końcowy praktyczny poprawkowy rozwiązania zadań

35.7 Końcowy sprawdzian teoretyczny (test).

Sprawdzian końcowy teoria - końcowy sprawdzian teoretyczny

rozwiązania - końcowy sprawdzian teoretyczny

Sprawdzian końcowy teoria poprawa – sprawdzian końcowy teoria poprawa

końcowy sprawdzian teoretyczny poprawkowy - rozwiązania

Sprawdzian końcowy teoria poprawa – sprawdzian końcowy ostatniej szansy teoria poprawa

końcowy sprawdzian teoretyczny poprawkowy ostatniej szansy - rozwiązania

Rozwiązania niektórych zadań – rozwiązania link