Wprowadzenie do edycji makr w Excelu

Zakładam, że choć teoretycznie wiesz coś na temat programowania.
Dobrze by było, gdybyś w tej chwili miał otwartego Excela . Będę omawiał pewne rzeczy, które mógłbyś testować od razu. Jeśli znasz VBA to nie czytaj. To będą podstawy podstaw. Poczekaj lepiej na kolejne części.

Zakładam, że masz otwarty edytor VBA. Jeśli nie, to wciśnij proszę w Excelu ALT+ f11 – ten skrót nie zmienił się na szczęście w Excelu 2007.

Makra można wpisywać w kilku miejscach(o tym kiedy indziej). Nagrywanie makra powoduje automatyczne dodanie nowego modułu. My dodamy go ręcznie poprzez wybór w menu edytora „INSERT->MODULE „ Nagrałem przykładowe makro, żebyśmy mieli co omawiać . Oto jego wynik, który możesz wkleić do edytora :

```
Sub Makro1()
'
' Makro1 Makro
'
'
ActiveCell.value = "ala ma kota"
Range("A1").Select
Selection.Copy
Range("H12").Select
ActiveSheet.Paste
Rows("11:11").Select
Selection.EntireRow.Hidden = True
Columns("B:B").Select
Selection.FormulaR1C1 = "1"
Sheets.Add After:=Sheets(Sheets.Count)
End Sub
```

Po wciśnięciu klawisza f8 kolejne kroki są wykonywane pojedynczo. Excel dopuszcza edytowanie makra w pewnym zakresie pomiędzy wciśnięciami f8. Możesz także ustawiać znacznik wykonywania w dowolnym punkcie programu , pomijać jego części, lub powracać do wcześniejszych fragmentów. Jest więcej opcji, ale nie zdążę wszystkich przedstawić.

Takie makro możemy w prawdzie wykorzystać, ale nie przyda nam się na wiele. Ktoś otworzy ważny plik, zobaczy, że jest makro o nic nie mówiącym tytule, odpali je i zniszczy ważne dane (dobrze, że makro nie zapisuje pliku). Zaczniemy więc od nazwy.

Sub - określa nam jedną procedurę – makro, czyli listę poleceń do wykonania. Taki blok kończy się słowami **„end sub „** Nazwy w VBA mogą składać się z liter, cyfr i podkreślników, jednak zaczynać mogą się tylko od liter. To, co jest w nawiasach to parametry- na chwilę to zostawimy. Powiem tylko, że jeśli są jakieś parametry, to makro nie będzie widoczne w liście makr(ale nie tylko wtedy). Zmień nazwę makra

W kolejnych kilku linijkach są komentarze. Wszystko w linijce po znaku ‘ jest komentarzem. W tym wypadku warto by było wpisać po nazwie makra krótki jego opis. VBA nie posiada czegoś takiego jak komentarze blokowe. Oznacza to ,że jeśli chcesz pisać komentarz w kilku linijkach na początku każdej będziesz musiał dodać znak apostrofu.

Zanim przejdziemy dalej, chciałbym zwrócić Twoją uwagę na drugie użycie komentarzy. W mojej pracy bywa tak, że muszę na chwilkę wyłączyć część działającego kodu. Ponieważ nie mam do dyspozycji komentarzy blokowych, muszę wstawić na początku każdej linijki, którą chcę chwilowo wyłączyć apostrof. Mógłbym robić to ręcznie, ale edytor może zrobić to za nas. Klikamy więc VIEV-> TOOLBARS->EDIT. Pasek ,który się pojawił powinien być, moim zdaniem, domyślnie włączony. Na początku mojej pracy z Excelem wstawiałem apostrofy ręcznie, co było uciążliwe. Na pasku masz przycisk Comment Block, z pomocą którego możesz zakomentować zaznaczony fragment kodu.

Przyjrzymy się pierwszej linijce z kodem. Jest to :

```
ActiveCell.value = "ala ma kota"
```

Oznacza to tyle co ustaw wartość w aktywnej komórce na słowa „Ala ma kota” Jeśli znasz język angielski, czytanie kodu prostych makr powinno być dla ciebie dość łatwe.

ActiveCell to aktywna komórka. Jeśli odpalisz makro, zdanie „Ala ma kota” pojawi się zawsze w komórce , którą miałeś zaznaczoną. Możesz to przetestować.
Do składowych obiektu, czy to składnika (czyli zmiennej) , czy metody (czyli funkcji składowej) odwołujemy się za pomocą kropki. Jeśli wydało ci się to skomplikowane, to pomyśl po prostu, że jeśli chcesz coś zrobić z obiektem to używasz do tego kropki.
W kolejnych wierszach także widzimy użycie kropki, czyli robimy coś na obiektach:

```
Range("A1").Select ‘ zakres „A1” zaznacz
Selection.Copy ‘ Zaznaczenie skopiuj
Range("H12").Select ‘ zakres „h12” zaznacz
ActiveSheet.Paste’ W aktywnym arkuszu wklej ( po prostu wklej tu gdzie jesteś )
Rows("11:11").Select ‘zaznacz rzędy od 11 do 11
Selection.EntireRow.Hidden = True ‘ W zaznaczeniu ,wybierz wszystkie rzędy i ustaw wartość ukryte na prawda.
'.....
```

I tak dalej .

Nawet jeśli nie rozumiesz wszystkiego , nie przejmuj się . Gdyby nauka przychodziła nam bez trudu , to pewnie wszyscy kochalibyśmy szkołę i byli geniuszami.
Korzystaj z dostępnej w edytorze pomocy, jest dobrze napisana, choć na początku może sprawiać spore problemy. Po pewnym czasie powinieneś z łatwością odnajdywać to, czego szukasz i będziesz rozumiał skomplikowane na pierwszy rzut oka podpowiedzi.

Wróćmy jednak do naszego makra. Nagrywając je chciałem ułatwić sobie wpisywanie tekstu w pierwszych dziesięciu polach (od A1 do A10). Niestety, ponieważ jestem nie doświadczony, po drodze wykonałem sporo różnych zbędnych operacji, a makro nie robi tego, co chciałem. Niestety często tak bywa. Jesteśmy zmuszeni do edytowania makra, i do wyciągnięcia z tego co się nagrało, tylko tego co jest nam potrzebne.
Analizując jeszcze raz kod, doszedłem do wniosku, że większość linijek jest zbędna i robią tylko bałagan. Skasować musiałem wszystko, oprócz pierwszej linijki, a i ta nie działa prawidłowo. Nie oznacza to, że niczego się nie nauczyłem. Poznałem podstawowe obiekty, takie jak:

Range – Zakres
Sheet – Lista Arkuszy
ActiveSheet – Aktywny Arkusz
Selection – zaznaczenie
Columns – kolumny

Ta wiedza przyda nam się w kolejnych makrach. Nagrywanie makr to dobry sposób nauki. Widzimy co krok po kroku zrobił program. Trzeba jednak być bardzo ostrożnym, żeby nie nagrywać niepotrzebnych rzeczy i nie zaciemniać sobie kodu.

Chciałem mieć w polach a1 do a10 napis „Ala ma kota”. Została mi tylko linijka :
ActiveCell.value = "ala ma kota"

No i co? Będę zaznaczał kolejne komórki i uruchamiał makro? To ja dziękuję za taką pomoc! Przypadek, który omawiamy jest trywialny i szybciej wykonalibyśmy go „na piechotę” (np. zaznaczając komórki, wpisując „Ala ma kota” i wciskając ctlr+enter), ale na razie się uczymy. Najkrótszy kod, którym moglibyśmy rozwiązać ten problem to :

Range(„a1:a10”).value = „Ala ma kota”

Przetestuj. Czy to działa? Jest sukces!
To co? Kończymy? O nie! Pomęczę was jeszcze. Nagle zachciało mi się, żeby w kolejnych linijkach wpisana była liczba przed zdaniem, i żeby te liczby były 2 razy większe niż numer wiersza.

Będziemy musieli zmodyfikować nasze makro.
Masz już podstawową umiejętność edycji makr. Wiesz, że z obiektami możesz robić różne rzeczy. Im więcej będziesz poznawał obiektów, tym więcej rzeczy będziesz w stanie wykonać.

Czego nam brakuje? Zmiennych, pętli i instrukcji warunkowych.

W tak krótkim tekście nie poznamy wszystkiego, ale chcąc zarazić cię tematem pokażę wykorzystanie jednej pętli i jednej tylko zmiennej.
Orientujesz się pewnie, co to są zmienne. Wiesz coś o typach zmiennych, albo o deklaracji zmiennych?
Jeśli nie, to na razie nie szkodzi. Choć dobrą praktyką jest deklarowanie zmiennych (moim zdaniem), to w VBA nie musisz tego robić. Nie musisz też definiować ich typu (choć to

pomaga przy obiektach). Co to oznacza? Że jeśli chcesz użyć zmiennej to po prostu jej używasz.
Przykład:

X = 2

Takiej zmiennej możesz użyć jako tekstu, lub jako liczby (ma typ variant)
Modyfikować zmienne możesz w ten sam sposób

X= X+1

Czyli jeśli x= 1 to po wykonani tej linijki x będzie wynosić 2

Żeby rozwiązać nasz problem będę musiał zrobić coś takiego:

Licząc od jednego do dziesięciu wpisuj w kolejne wiersze numer liczby pomnożony przez 2 i „Ala ma kota”

Wykorzystamy pętlę for
For to taka pętla, którą przetłumaczyłbym jako:” Dla zmiennych, od pewnej wartości do kolejnej, wykonaj zadane polecenia.”
To niefortunne tłumaczenie, dla mnie jednak prościej wygląda w kodzie. Na przykład :

```
For x = 1 to 10
‘ Tu będą instrukcje
Next x
```

Taka pętle zaczyanmy od słowa for , poźniej "mówimy" komputerowi od ilu do ilu ma liczyć, a następnie pokazujemy mu , gdzie kończą się instrukcje które ma wykonać (next x)
Czyli komputer policzy od 1 do 10 i wykona tyle razy to co jest w środku.
Tego potrzebujemy! Pozostaje nam wydrukować wartości. Po wstawieniu naszego kodu drukującego mamy takie coś:

```
For x = 1 to 10
ActiveCell.value = "ala ma kota"
Next x
```

Jak dodać do tego numer wiersza razy 2? Wszystkie teksty w VBA łączymy za pomocą znaku & (pamiętamy o dodaniu spacji) . Czyli nasz kod mógłby wyglądać tak:

```
ActiveCell.value = x & " ala ma kota"’ wydrukuj x i spację i „Ala ma kota”
```

Wystarczy, że pomnożymy x*2 i będziemy mieli to, o co nam chodziło:

ActiveCell.value = x *2 & " ala ma kota"' wydrukuj x*2 i spację i „Ala ma kota”

[to jest nie eleganckie rozwiązanie]

Jeśli puścisz makro instrukcja po instrukcji (F8), zauważysz, że makro drukuje 10 razy tekst w pierwszej linijce. Brakuje nam przejścia do kolejnej linijki. Przy pierwszym podejściu zrealizujemy to za pomocą metody offset.
Offset to taka funkcja , która wskazuje na komórkę przesuniętą , o pewną liczbę komórek w danym kierunku. Przykład zilustruje to lepiej.

selection.offset(0,1).select

zaznaczy komórkę w prawo o jeden od zaznaczonej. Pierwsza liczba przesuwa w dół, druga w lewo. (w górę i w prawo można przesuwać wpisując liczby ujemne)
My chcemy komórkę w dół więc nasz kod będzie wyglądał tak:

```
For x = 1 to 10
ActiveCell.value = x *2 & " ala ma kota"
selection.offset(1,0).select
Next x
```

Działa! Tyle ,że jeśli przed uruchomieniem, zaznaczę inną komórkę niż a1, to znów mam błąd.

Mógłbym zaznaczyć za pomocą makra a1, ale wolałbym inaczej wskazywać na komórki.

Pomoże nam słowo cells (komórki).

Cells wskazuje na komórkę, na podstawie liczb. Na przykład cells(1,1) wskazuje na a1 , a cells(1,2) na b1.
Już wiesz jak będzie wyglądał nasz finalny kod? Spróbuj napisać go samemu.

Jeśli udało Ci się napisać samemu to super
powinno to wyglądać mniej więcej tak :

```
For x = 1 to 10
Cells(x,1) = x *2 & " ala ma kota"
Next x
```

Każdy z nas powinien dbać o formę. W natłoku zdarzeń czasem trudno jest nam na to znaleźć czas, ale warto się starać. Dziś porozmawiamy o formach i zdarzeniach w VBA. Zapraszam do „czwartej i ostatniej części trylogii” o makrach w Excelu ;)

Wcześniej napisaliśmy prostą funkcję tworzącą log wejść na nasz nowy raport. Nie wiem,czy pamiętasz, ale umieściliśmy ją w module Ten_skoroszyt, w makrze Workbook_Open() .

Makro Workbook_Open odpowiada za obsługę otwierania skoroszytu. Zobaczmy zatem edytor makr i sprawdźmy, na jakie inne zdarzenia możemy zareagować.

W oknie eksploratora Projektu VBA, klikamy podwójnie na obiekt "ten_skoroszyt" i zmieniamy (general) na workbook w polu wyboru. Po prawej mamy listę dostępnych opcji. Aby obsłużyć jakieś zdarzenie wybieramy je z listy, a następnie wpisujemy nasz kod w wygenerowany „szkielet makra". W liście dla zeszytu mamy np. makro before save (przed zapisem)

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
End Sub
```

zwróć uwagę na parametry makra SaveAsUI mówi nam czy zostanie wyświetlone okno „zapisz jako" ,a Cancel mówi nam czy zapisywanie zostanie anulowane. Zapewne dostrzegłeś także słowo BYVAL , oznacza ono, że zmienna SaveASUI została przekazana przez wartość i jej ewentualna zmiana na nic nie wpłynie(zagadnienie przekazywania zmiennych do funkcji wykracza poza ten wpis. Obiecałem się streszczać :)) Cancel nie ma takiego przedrostka i możemy zmienić jego wartość. Zróbmy to!

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
cancel = true
End Sub
```

i zapiszmy naszą pracę..... ups. Nie da się! Przed zapisaniem do programu zostaje przekazana wartość zmiennej Cancel(anuluj) i zapis zostaje anulowany. My powinniśmy móc dokonać zmian, sprawmy więc, żeby nikt , poza naszym użytkownikiem, nie mógł zapisać tego pliku.

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
If Not Application.UserName = "molexor" Then ' wpisz własny username
Cancel = True
End If
End Sub
```

teraz żaden użytkownik nie zapisze tego pliku (chyba że wyłączy obsługę makr).

Poeksperymentuj troszkę ze zdarzeniami, jest ich naprawdę sporo, od otwarcia pliku , po zmianę selekcji arkuszy, czy przeliczanie zeszytu. Warto się z nimi zapoznać.

Przykładowe praktyczne zastosowanie:

raport, o którym mówiliśmy w poprzedniej części, jest bardzo rzadko odwiedzany. Jego uzupełnienie nie trwa długo i także zautomatyzowaliśmy je makrem, ale ponieważ raport jest mało istotny, nie pamiętamy o jego uzupełnianiu. Wykorzystując potęgę makr jesteśmy w stanie zaradzić temu problemowi. Kiedy szef otwiera raport, wywoływane jest makro

uzupełniające dane, które samo sprawdza,czy raport jest uzupełniony i w razie konieczności naprawia to karygodne zaniedbanie. Wszyscy są szczęśliwi.

Zdarzenia zeszytu nie są jedyni na które możemy reagować. Klikając podwójnie w eksploratorze projektu w dowolny arkusz i wybierając pozycję Worksheet w polu wyboru , zobaczymy listę dostępnych opcji dla arkusza. Zajmiemy się tylko jednym dla przykładu. Będzie to :

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
End Sub
```

Zmienna target to miejsce/komórka kliknięcia, a za pomocą zmiennej Cancel możemy powstrzymać program od domyślnego działania, którym jest w tym wypadku rozpoczęcie edycji komórki.

Weźmy przykładową tabelkę z ocenami końcowymi uczniów (liczby są losowe)

Imię i nazwisko	matematyka	j. polski	historia	geografia	biologia
Jan Kowalski	5	4	4	4	5
Piotr Kowalski	4	4	5	4	6
Halina Kowalska	1	5	5	5	3
Euzebiusz Kowalski	1	3	1	5	5
Dezyderiusz Kowalski	2	3	1	3	6
Izabela Kowalska	4	5	6	3	4
Banaś Michał :P	4	4	4	5	2

Chciałbym, żeby po podwójnym kliknięciu na ocenie pokazało nam się okienko z ocenami ucznia z przedmiotu, który wskazaliśmy klikając. Dzięki obsłudze zdarzeń jesteśmy w stanie zrealizować coś takiego w bardzo prosty sposób

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
If Target.Row > 2 And Target.Row < 10 And Target.Column > 2 And Target.Column < 8
Then
oceny = pobierzoceny(Cells(2, Target.Column).Value, Cells(Target.Row, 2).Value)
strocen = "Oceny z przedmiotu: " & Cells(2, Target.Column).Value & Chr(10) & _
"Ucznia: " & Cells(Target.Row, 2).Value & Chr(10) & "To: "
For Each el In oceny
strocen = strocen & el & " , "
Next el

MsgBox strocen
Cancel = True
End If
End Sub
Function pobierzoceny(przedmiot, nazwiskoImie)
```

```
'zrezygnowałem z jakiejkolwiek logiki w funkcji. trzeba ją dopisać
pobierzoceny = Array(1, 2, 3, 4, 5, 6)
End Function
```

za pomocą rozciągniętego „ifa” określiliśmy dla których kolumn i rzędów będziemy działać. Zmienna target ma typ Range, czyli z angielskiego zakres, stąd właściwości row i column(wiersz i kolumna) funkcja pobierzocenmy powinna wyglądać nieco inaczej, jednak na potrzeby naszego przykładu wystarczy taki mały „implant” .

Podobnie obsługuję się zdarzenia w formach i kontrolkach ActiveX. Po dodaniu takiej kontrolki AX na arkusz, w edytorze będziemy mieli do wyboru zdarzenia z nią związane (do wyboru obok worksheet).

Proponuję żebyś drogi czytelniku dodał sobie kilka kontrolek i spróbował je obsłużyć . Wprowadzenie kontrolek ActiveX pozwoli ci na tworzenie już całkiem ciekawych aplikacji w Excelu. Pamiętaj że kontrolki takie jak combobox należałoby w którymś momencie zainicjalizować listą wyboru. Realizacja zależy już od danej aplikacji.