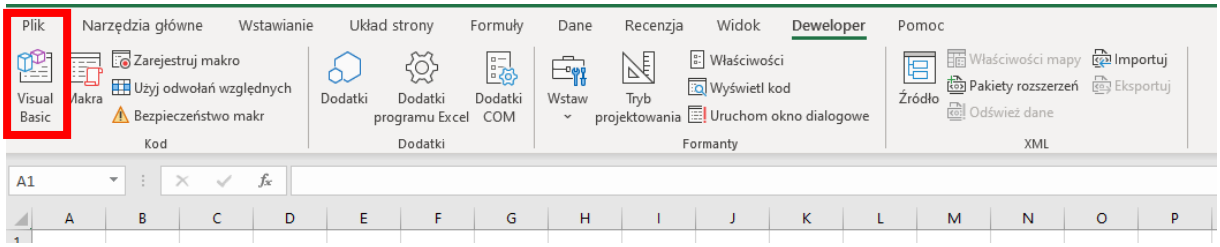
Arkusz kalkulacyjny – opcje zaawansowane (edycja komórek i wypełnianie seryjne, edycja formuł, przegląd funkcji, wykorzystanie pilota danych, makrodefinicje), cz. 4

Zad. 1

Na karcie *Developer* w Excelu (grupa *Kod*) wybierz opcję *Visual Basic* (rys. 1). Po otwarciu edytora pojawi się okno zintegrowanego środowiska programistycznego języka Visual Basic (Integrated Development Environment – IDE; rys. 2).



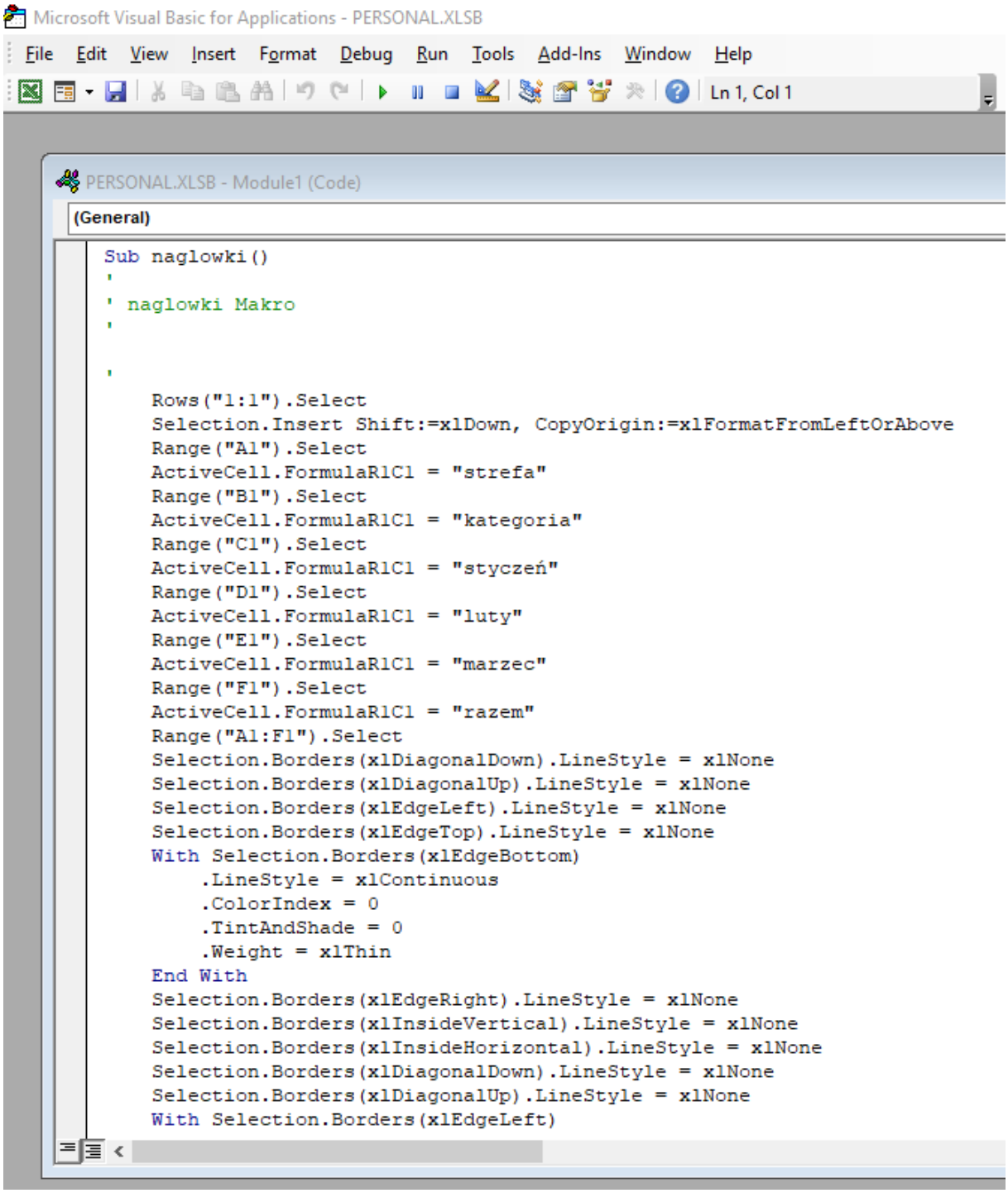
Rys. 1

Zad. 2

Z menu *View* na pasku narzędzi wybierz opcję *Project Explorer* (*Ctrl + R*) i *Properties Window* (*F4*; rys. 3). Otwarte okno edytora wzbogaci się wówczas o eksploratora projektu oraz okno właściwości. Kliknij prawym przyciskiem myszy nazwę projektu *VBAProject* (*Zeszyt 1*) i z menu kontekstowego wybierz opcję *Insert -> Module* (rys. 4). Pojawi się wówczas okno modułu (rys. 5). Dobrym zwyczajem jest najpierw zapisanie całego projektu. Wybierz skoroszyt *Arkusz1* projektu *VBAProject* klikając na nazwie skoroszytu lewym przyciskiem myszy. Następnie z menu *File* wybierz opcję *Save Zeszyt1* (*Ctrl + S*), zmień nazwę arkusza na *mój_arkusz1* i zapisz go w folderze grupowym zmieniając typ pliku na *Skoroszyt programu Excel z obsługą makr* (rys. 6-8).

Zad. 3

Na rys. 9 przedstawiono proste makro przypisujące określonym wartościom etykiety tekstowe „Dodatni” lub „Ujemny” w zależności od tego, czy wartość jest dodatnia, czy ujemna. Jest to jednocześnie procedura rozpoczynająca się od słowa kluczowego *Sub*. Procedurą możemy określić najmniejszą cześć kodu, która posiada własną nazwę. Procedura to także najmniejsza część kodu, która może być uruchomiona niezależnie od innej części utworzonego kodu. Procedura *Sub* służy do automatyzacji określonych, powtarzalnych działań. Przyjmuje ona argumenty, ale nie zwraca wartości, dlatego nie może być wykorzystywana w formułach. Procedurę deklaruje się właśnie za pomocą słowa kluczowego *Sub*. Jest ona zamykana przez instrukcję *End Sub*. Przepisz makro pokazane na rys. 9 do nowo otwartego okna modułu. Następnie w arkuszu *Arkusz1* skoroszytu *mój_arkusz1* wpisz dane pokazane na rys. 10. Uruchom następnie utworzone przed chwilą makro (*If_Loop*).

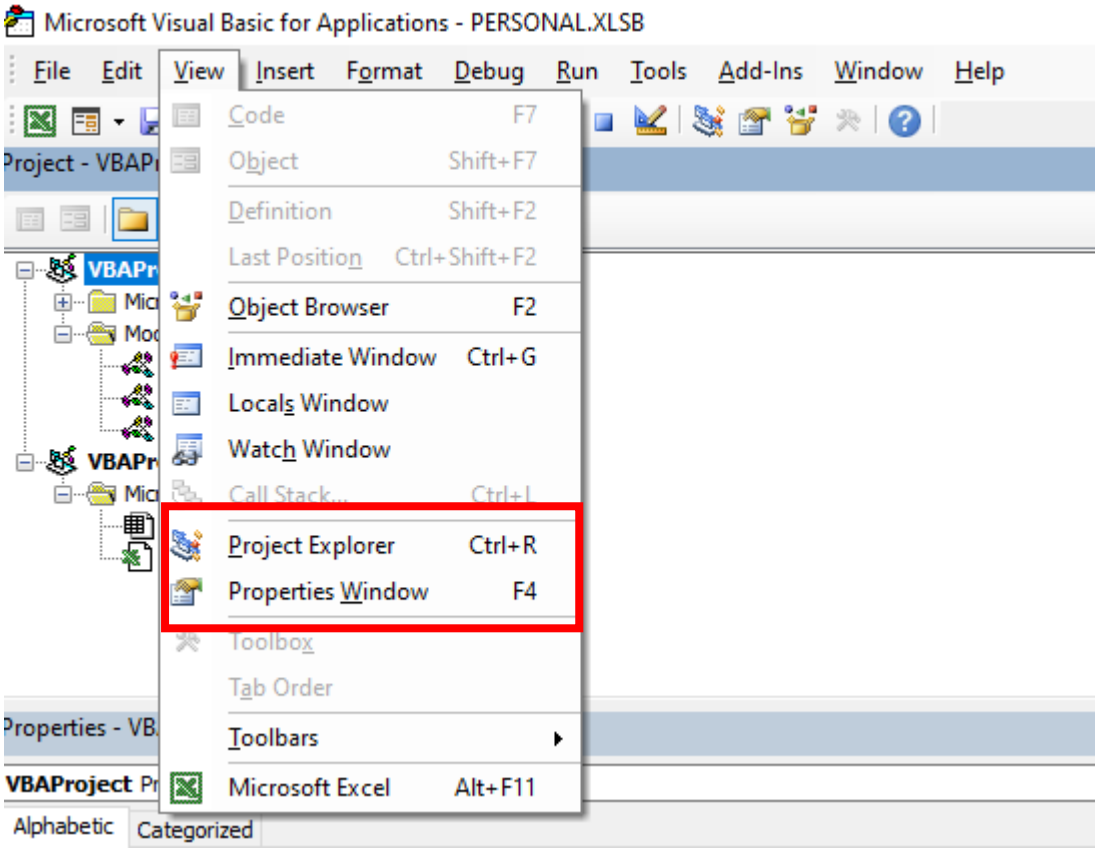


Rys.2

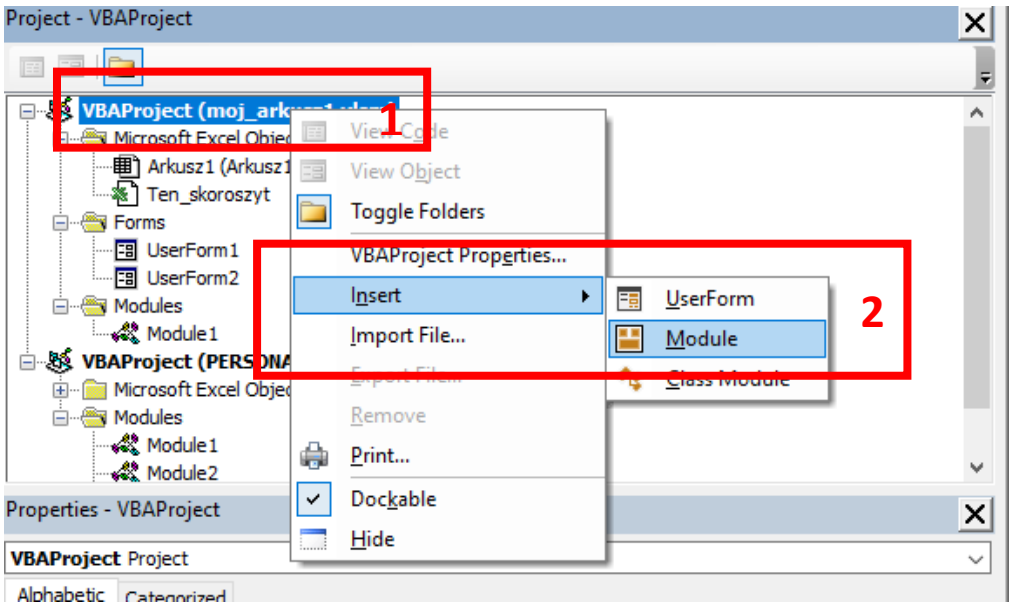
Zad. 4

Instrukcje *If* w VBA (Visual Basic for Applications) pozwalają na sprawdzanie, czy określone wyrażenia są prawdziwe, czy fałszywe, wykonując następnie różną część kodu w zależności od wyniku, np.:

If Range("a2").Value > 0 Then Range("b2").Value = "Dodatni"

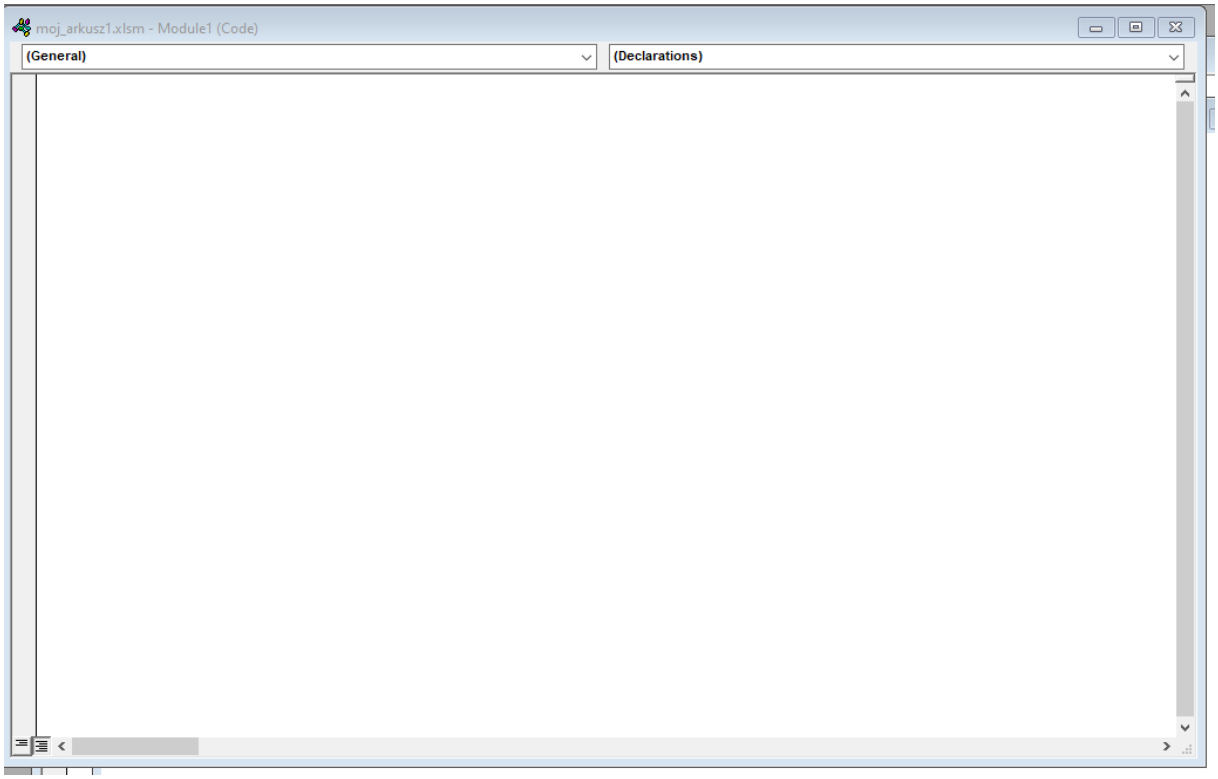


Rys. 3

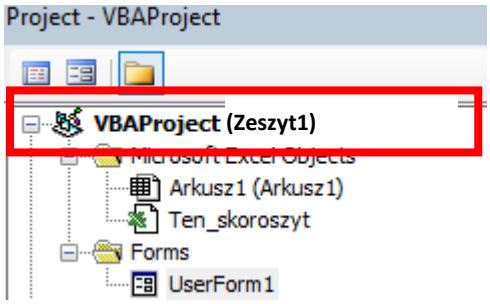


Rys. 4

Powyższa instrukcja sprawdza, czy wartość komórki A2 jest większa od 0. Jeśli tak, to komórka B2 przyjmuje wartość *Dodatni*. Przepisz makro pokazane na rys. 11 zapisując je w nowym module (*Insert -> Module*; zwróć uwagę na podkreślnik poprzedzony bezpośrednio pojedynczą spacją po słowie kluczowym *Then* – przenosi on instrukcję do kolejnego wiersza). Następnie przepisz dane pokazane na rys. 12 do nowego arkusza (*Arkusz2*) w skoroszycie *mój_arkusz1*. Uruchom makro (*IF_Then*).



Rys. 5



Rys. 6

Zad. 5

Pamiętaj, że przy sprawdzaniu warunków używamy operatorów porównania =, >, <, <>, <=, >=. Poniżej przedstawiono składnię prostej jednowierszowej instrukcji *If*:

If [sprawdzane_wyrażenie] Then [akcja]

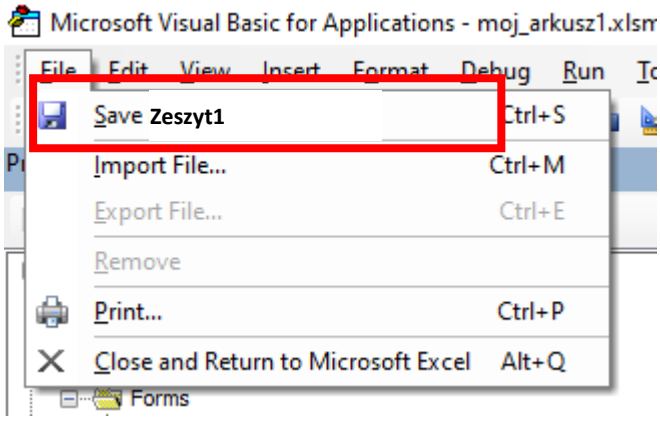
Aby uczynić ją bardziej przejrzystą, możesz wykorzystać znak przeniesienia do nowego wiersza (podkreślnik), aby rozbić instrukcję *If* na dwa wiersze (jak to zostało zrobione wcześniej):

```
If [sprawdzane_wyrażenie] Then _
   [akcja]
If Range("a2").Value > 0 Then _
   Range("b2").Value = "Dodatnia"
```

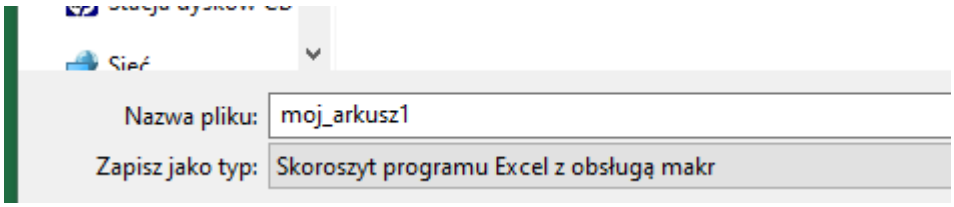
Powyższa jednowierszowa instrukcja *If* zdaje egzamin, kiedy sprawdzasz tylko jednej warunek. Jeżeli natomiast instrukcje *If* stają się bardziej skomplikowane i obejmują większą liczbę warunków, konieczne będzie dodanie instrukcji *End If* na końcu bloku *If...Then...Else*:

```
If Range("a2").Value > 0 Then
   Range("b2").Value = "Dodatni"
End If
```

Przepisz makro pokazane na rys. 13 zapisując je w nowym module (*Insert -> Module*). Następnie przepisz dane pokazane na rys. 14 do nowego arkusza (*Arkusz3*) w skoroszycie *mój_arkusz1*. Uruchom makro (*IF_Then*).



Rys. 7



Rys. 8

moj_arkusz1.xlsm - Module1 (Code)

(General)

```
Sub If_Loop()
    For Each Cell In Range("A2:A6")
        If Cell.Value > 0 Then
            Cell.Offset(0, 1).Value = "Dodatnia"
        ElseIf Cell.Value < 0 Then
            Cell.Offset(0, 1).Value = "Ujemna"
        Else
            Cell.Offset(0, 1).Value = "Zero"
        End If
    Next Cell
End Sub
```

Rys. 9

	A	B
1	**Wartość**	**Grupa**
2	50	
3	-10	
4	0	
5	1	
6	-13	

Rys. 10

moj_arkusz1-2.xlsm - Module2 (Code)

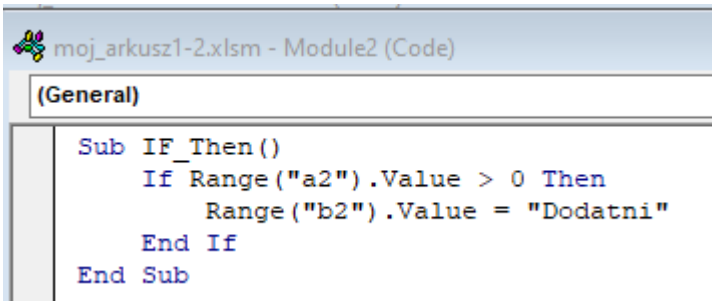
(General)

```
Sub IF_Then()
    If Range("a2").Value > 0 Then _
    Range("b2").Value = "Dodatni"
End Sub
```

Rys. 11

	A	B
1	**Wartość**	**Grupa**
2	50	

Rys. 12



Rys. 13

	A	B	C
1	**Wartość**	**Grupa**	
2	40		
3			

Rys. 14

Źródła:

https://www.vbtutor.net/lesson2.html

https://www.automateexcel.com/vba/else-if-statement#VBA_If_Statement

https://www.cognity.pl/kurs-vba-procedura-sub-i-procedura-function,blog,153.html

https://learn.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/if-then-else-statement