

PROGRAMOWANIE W VBA NA ZAJĘCIACH FAKULTATYWNYCH Z INFORMATYKI

Mirosława Kołowska-Gawiejnowicz

Wiktoria Zborowska

Wydział Matematyki i Informatyki

Uniwersytet im. Adama Mickiewicza w Poznaniu

ul. Uniwersytetu Poznańskiego 4, 61-614 Poznań

mkolowsk@amu.edu.pl, wikzbo1@st.amu.edu.pl

Abstract. *The aim of the paper is to show that certain points of the computer science curriculum can be implemented by teaching Visual Basic for Applications. The presentation is illustrated with examples in VBA, showing how the considered algorithms can be implemented in this language. In addition, a proposal for tasks for optional computer science classes is presented.*

1. Wstęp

Umiejętność programowania, która rozwija u uczniów umiejętności myślenia komputacyjnego, jest jednym z głównych celów nauczania informatyki i ma miejsce na wszystkich etapach edukacji. Realizacji podstawy programowej w zakresie programowania w szkole podstawowej służą w młodszych klasach języki wizualne, a w starszych języki imperatywne, np. Python. Nauka programowania kontynuowana jest w szkole ponadpodstawowej z wykorzystaniem języków Python lub C++. Pewne wymagania podstawy programowej z informatyki można zrealizować w trakcie nauki języków nie tylko imperatywnych, ale i deklaratywnych. W [2] pokazano to na przykładzie języka funkcyjnego Haskell, a w [3] na przykładzie języka programowania w logice - Prolog.

Celem niniejszego artykułu jest pokazanie, że programowania można nauczać również używając Visual Basic for Applications (VBA). Podamy parę przykładów implementacji w VBA algorytmów wymienionych w podstawie programowej. VBA może być wybrany jako narzędzie do programowania gier i rozwiązywania bardziej złożonych zadań w ramach zajęć fakultatywnych z informatyki.

2. Krótka charakterystyka języka VBA

W 1963 roku powstał język programowania BASIC (Beginners All-Purpose Symbolic Instruction Code), który miał umożliwić pisanie programów osobom niezajmującym się zawodowo programowaniem. W 1991 roku, po pojawieniu się systemu Windows, powstała wersja VB (Visual Basic), czyli BASIC połączony ze środowiskiem programowania w tym systemie. Na bazie tego języka oparty został VBA (Visual Basic for Applications), czyli VB dołączony do aplikacji Microsoft Office (takich jak Excel, Access, Word), którego zadaniem było ułatwienie użytkownikom samodzielnego rozszerzania tych programów poprzez tworzenie tzw. makr, czyli list instrukcji, które dana aplikacja może automatycznie wykonać.

Program w VBA jest podzielony na procedury i funkcje. Procedury rozpoczyna się i kończy odpowiednio słowami kluczowymi Sub i End Sub, natomiast funkcje – słowami kluczowymi Function i End Function.

Zmienne w VBA deklaruje się za pomocą słowa kluczowego Dim. Podstawowe typy danych to m.in. Integer, Long, Double, String, Boolean. Na uwagę zasługuje typ uniwersalny Variant, ponieważ zmienna tego typu może przechowywać wartość dowolnego typu.

VBA operuje *obiektami*, udostępnia około 100 klas obiektów. Przykładami obiektów są: skoroszyty (Workbook), arkusze (Worksheet), zakresy (Range), komórki (Cells), wykresy (Chart), tabele przestawne (PivotTable).

Obiekty mają *właściwości*. Właściwość może być traktowana jako *wartość pola* obiektu. Np. Obiekt Range ma właściwości takie jak Value (wartość) i Name (nazwa). Obiekt Chart ma właściwości takie jak HasTitle (ma tytuł) i Type (typ). Odwołanie się do właściwości wymaga podania obiektu i jego właściwości oddzielonych kropką. Np. Odwołanie się do wartości komórki A1 w Arkuszu1:

Worksheets("Arkusz1").Range("A1").Value

Obiekty mają *metody*. Metoda to działanie wykonywane przez obiekt. Np. jedną z metod obiektu Range jest ClearContents. Metoda ta usuwa zawartość komórek z zakresu. Inne przykłady metod: dla obiektu Workbook: Save (zapisuje skoroszyt), Close (zamyka skoroszyt), dla obiektu Worksheet: Delete (usuwa arkusz), Activate (aktywuje arkusz), dla obiektu Range: Copy (kopiuje zakres), Select (zaznacza zakres).

VBA oferuje różne rodzaje pętli: For... Next (z wariantami), For Each..., Do While...Loop, Do Until... Loop, Do ... Loop While ..., Do... Loop Until....

Instrukcje warunkowe w VBA: If ...Then... End If, If... Then Else... End If, Select Case... .

Wymienione powyżej podstawowe elementy języka VBA [1], [4], [5] są wystarczające do zastosowań szkolnych. W poniższej tabeli zestawiono je z elementami języka Python w celu wskazania podobieństw, które mogą być użyteczne w poznawaniu Visual Basica po wcześniejszej nauce języka programowania Python [6].

	VBA	Python
Definiowanie funkcji	Function Nazwa(argument1 As Typ, ...) As Typ Instrukcje Nazwa = wynik End Function	def Nazwa(argumenty): Instrukcje return
Definiowanie procedur	Sub Nazwa(argumenty As Typ) Instrukcje End Sub	def Nazwa(argumenty): Instrukcje
Instrukcja warunkowa	If warunek Then Instrukcje1 Else Instrukcje2 End If	if warunek: Instrukcje1 else: Instrukcje2
Pętla For	For licznik = wartośćpoczątkowa To wartośćkońcowa Step krok Instrukcje Next licznik	for licznik in zakres: Instrukcje
Podstawowe typy danych	Integer Double String Boolean Variant	int float str bool brak typu uniwersalnego
Operatory arytmetyczne	a+b, a-b, a*b, a/b, a\b, a Mod b, Sqr(a), a^b	a+b, a-b, a*b, a/b, a//b, a%b, sqrt(a), a**b
Operatory porównania i logiczne	a = b, a <> b, a > b, a < b, a >= b, a <= b, And, Or, Not	a == b, a != b, a > b, a < b, a >= b, a <= b, and, or, not
Wczytywanie danych	InputBox()	input()
Wyświetlanie danych	MsgBox()	print()

Tabela 1. Zestawienie elementów języka VBA i Python

3. Algorytmy w podstawie programowej

Przypomnijmy wybrane wymagania szczegółowe podstawy programowej z informatyki z zakresu podstawowego i rozszerzonego dotyczące rozwiązywania problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji [7]:

Algorytmy wyszukiwania i sortowania	porządkowanie ciągu liczb: przez wstawianie i metodą bąbelkową; sortowanie ciągu liczb przez scalanie; sortowanie szybkie; znajdowanie określonego elementu w zbiorze uporządkowanym metodą binarnego wyszukiwania; jednocześnie wyszukiwanie elementu najmniejszego i największego; wyszukiwanie elementów liniowe i przez połowienie (do znajdowania elementów w zbiorze, sortowania przez wstawianie); znajdowanie w ciągu podciągów o różnorodnych własnościach, np. najdłuższego spójnego podciągu niemalejącego, spójnego podciągu o największej sumie; metoda dziel i zwyciężaj (jednocześnie znajdowanie minimum i maksimum, sortowanie przez scalanie i szybkie)
Algorytmy numeryczne	wyznaczanie miejsc zerowych funkcji metodą połowienia / przybliżonego rozwiązywania równań; obliczanie przybliżonej wartości pierwiastka kwadratowego; obliczanie wartości wielomianu za pomocą schematu Hornera; szybkie potęgowanie liczb w wersji iteracyjnej i rekurencyjnej; metoda połowienia
Algorytmy teorii mnogościowe	badanie pierwszości liczby; zamiana reprezentacji liczb między pozytywnymi systemami liczbowymi; działania na ułamkach z wykorzystaniem NWD i NWW; algorytm Euklidesa w wersji iteracyjnej i rekurencyjnej wraz z zastosowaniami; generowanie liczb pierwszych metodą sita Eratostenesa; rozkładanie liczby na czynniki pierwsze; wykonywanie działań na liczbach w systemach innych niż dziesiętny
Przetwarzanie tekstów	porównywanie tekstów; wyszukiwanie wzorca w tekście metodą naiwną; szyfrowanie tekstu metodą Cezara i przestawieniową
Algorytmy zachłanne i dynamiczne	wydawanie reszty najmniejszą liczbą nominałów; podejście zachłanne
Ciągi, rekurencja i inne techniki	obliczanie wartości elementów ciągu metodą iteracyjną i rekurencyjną, w tym wartości elementów ciągu Fibonacciego; rekurencyjne tworzenie fraktali; zamiana wyrażenia na postać w odwrotnej notacji polskiej i obliczanie jego wartości na podstawie tej postaci; rekurencja (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali)

Tabela 2. Algorytmy wskazane w podstawie programowej z informatyki

Wszystkie wymienione powyżej algorytmy można zaimplementować w VBA, który jest językiem imperatywnym i który z powodzeniem może zastąpić w nauce programowania inne języki programowania. W rozdziale 4 przedstawimy implementacje w VBA wybranych ww. algorytmów.

4. Przykłady procedur w VBA

Przykład 1. Algorytm Euklidesa. Już w szkole podstawowej omawiany jest algorytm Euklidesa znajdowania największego wspólnego dzielnika (NWD) dwóch liczb. Uczniowie klas 4-6 implementują ten algorytm z wizualnym języku programowania, np. w Scratchu, uczniowie klasy 8 programują w Pythonie, Poniżej prezentujemy algorytm Euklidesa w Pythonie i VBA w celu pokazania podobieństw obu kodów [6].

Implementacja w Pythonie:

```
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))
if a>0 and b>0:
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a
    print("NWD = ", a)
else: print("Proszę wprowadzić liczby naturalne.")
```

W VBA procedura realizująca algorytm Euklidesa jest następująca:

```
Sub NWD()
    a = Range("A2").Value
    b = Range("B2").Value
    If a > 0 And b > 0 Then
        Do While a <> b
            If a > b Then
                a = a - b
            Else b = b - a
            End If
        Loop
    Range("C2").Value = a
    Else Range("C2").Value = "Proszę wprowadzić liczby naturalne."
    End If
End Sub
```

Przykład 2. Algorytm badania pierwszości liczby. Funkcja CzyPierwsza sprawdza, czy podana jako argument liczba naturalna jest pierwsza i zwraca wartość logiczną True lub False. Procedura SprawdzPierwszoscLiczby wywołuje funkcję CzyPierwsza i w MsgBox wypisuje stosowny komunikat o pierwszości liczby. Implementację w VBA poprzedzamy, dla porównania, programem zapisanym w Pythonie [6]:

```
from math import floor, sqrt
def jest_pierwsza (n):
    m = floor(sqrt(n))
    for i in range(2,m+1):
        if n%i==0:
            return False
    return True
liczba = int(input("Podaj liczbę: "))
if liczba > 0 and jest_pierwsza(liczba):
    print("Liczba ", liczba, "jest pierwsza")
else:
    print("Liczba ", liczba, "nie jest pierwsza")
```

Implementacja w VBA:

```
Function CzyPierwsza(n As Long) As Boolean
Dim i As Long
If n < 2 Then
    CzyPierwsza = False
    Exit Function
End If
For i = 2 To Sqr(n)
    If n Mod i = 0 Then
        CzyPierwsza = False
        Exit Function
    End If
Next i
CzyPierwsza = True
End Function
```

```
Sub SprawdzPierwszoscLiczby()
Dim liczba As Long
Dim wynik As Boolean
liczba = InputBox("Podaj liczbę:")
If liczba <= 0 Then
```

```
MsgBox "Podano nieprawidłową liczbę."  
Exit Sub  
End If  
wynik = CzyPierwsza(liczba)  
If wynik Then  
    MsgBox "Liczba " & liczba & " jest liczbą pierwszą."  
Else  
    MsgBox "Liczba " & liczba & " nie jest liczbą pierwszą."  
End If  
End Sub
```

Przykład 3. Procedura realizująca sortowanie elementów tablicy metodą polegającą na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności, tzw. sortowanie bąbelkowe. Elementy wprowadzane są przez użytkownika za pomocą InputBox, a posortowane wypisane w wierszu 1 począwszy od kolumny A w arkuszu kalkulacyjnym.

```
Sub SortowanieB()  
    Dim n As Long  
    Dim tabl(1 To 100) As Double  
    Dim i As Long, j As Long  
    Dim pom As Double  
    n = InputBox("Podaj liczbę elementów n:")  
    For i = 1 To n  
        tabl(i) = InputBox("Podaj element nr " & i & " :")  
    Next i  
    For i = 1 To n - 1  
        For j = 1 To n - i  
            If tabl(j) > tabl(j + 1) Then  
                pom = tabl(j)  
                tabl(j) = tabl(j + 1)  
                tabl(j + 1) = pom  
            End If  
        Next j  
    Next i  
    For i = 1 To n  
        Cells(1, i).Value = tabl(i)  
    Next i  
End Sub
```

Przykład 4. Ciąg Fibonacciego. Funkcja FibRek oblicza i jako wartość zwraca n-ty wyraz ciągu Fibonacciego metodą rekurencyjną, funkcja FibIt – metoda iteracyjną.

```
Function FibRek(n As Long) As Long
    If n = 1 Or n = 2 Then
        FibRek = 1
    Else
        FibRek = FibRek(n - 1) + FibRek(n - 2)
    End If
End Function
```

```
Function FibIt(n As Long) As Long
    f1 = 1
    f2 = 1
    For i = 3 To n
        pom = f2
        f2 = f1 + f2
        f1 = pom
    Next i
    FibIt = f2
End Function
```

Przykład 5. Zamiana liczby z systemu binarnego na dziesiętny. Przedstawiamy dwa algorytmy zamiany liczby dwójkowej na dziesiętną. Niewielka modyfikacja procedur daje procedury zamieniające liczbę zapisaną w dowolnym systemie o podstawie n, gdzie $2 < n < 10$, na liczbę zapisaną w systemie dziesiętnym. Wykorzystane funkcje: Len – wartością jest liczba elementów ciągu, Mid(t,p,i) – wartością jest podciąg tekstu t od pozycji p o długości i.

```
Sub BinarnyNaDz1()
Dim b As String
Dim d As Long
Dim i As Integer
b = InputBox("Podaj liczbę binarną:")
d = 0
For i = 1 To Len(b)
    d = d * 2
    If Mid(b, i, 1) = "1" Then
        d = d + 1
    End If
Next i
End Sub
```

```
End If
Next i
MsgBox "Liczba dziesiętna: " & d
End Sub

Sub BinarnyNaDz2()
Dim b As String
Dim d As Long
Dim potega As Long
Dim i As Integer
b = InputBox("Podaj liczbę binarną:")
potega = 1
d = 0
For i = Len(b) To 1 Step -1
    If Mid(b, i, 1) = "1" Then
        d = d + potega
    End If
    potega = potega * 2
Next i
MsgBox "Liczba dziesiętna: " & d
End Sub
```

Przykład 6. Zamiana liczby z systemu dziesiętnego na binarny.

```
Sub DzNaBinarny()
Dim d As Long
Dim b As String
d = InputBox("Podaj liczbę dziesiętną:")
b = ""
Do While d > 0
    If d Mod 2 = 0 Then
        b = "0" & b
    Else
        b = "1" & b
    End If
    d = d \ 2
Loop
MsgBox "Liczba binarna: " & b
End Sub
```

5. Propozycja zajęć fakultatywnych z wykorzystaniem VBA

W czasie zajęć nieobowiązkowych, fakultatywnych z informatyki, nauczyciel ma pewną swobodę w planowaniu zajęć. VBA daje szeroki wachlarz możliwości zastosowań. Na lekcjach informatyki w zakresie rozszerzonym uczniowie rejestrują makra. Zapoznanie się z bezparametrowymi procedurami, czyli makrami VBA może być pierwszym krokiem z poznawaniu Visual Basic for Applications. Kolejnym – modyfikacja kodu zarejestrowanego już makra, a następnie przejście do samodzielnego programowania.

Zajęcia dodatkowe z informatyki wybierane są zazwyczaj przez uczniów chętnych poszerzyć swoją wiedzę i umiejętności. W tym celu można wykorzystać VBA:

- do tworzenia edukacyjnych aplikacji desktopowych, np. kalkulatora lub programów do nauki matematyki (lub innych przedmiotów),
- do projektowania gier edukacyjnych typu: quizy, łamigłówki.

Propozycje zadań na zajęcia fakultatywne:

Zadanie 1. Napisać grę: „Znajdź wylosowaną liczbę”.

Zadanie 2. Napisać zaawansowany kalkulator.

Zadanie 3. Napisać kalkulator dla liczb w zapisie rzymskim.

Zadanie 4. Napisać grę typu „Milionerzy” z pytaniami z matematyki, informatyki lub innej dowolnej dziedziny.

Zadanie 5. Napisać grę „Statki”.

Zadanie 6. Napisać grę „Kółko i krzyżyk”.

Zadanie 7. Napisać grę „15” polegającą na ułożeniu w kolejności rosnącej liczb od 1 do 15 w tablicy 4x4, począwszy od losowego umieszczenia tych liczb w komórkach tej tablicy, przesuwając liczby z zajętych komórek do komórek bez liczb.

Ad. Zadania 1. Poniżej procedury w VBA realizujące grę „Znajdź wylosowaną liczbę”. Procedura Losowanie odpowiada za wylosowanie liczby (użycie funkcji Rnd), wpisanie jej do komórki J1 i „ukrycie” poprzez ustalenie białego koloru czcionki. Procedura WybieranieLiczby sprawdza, czy w komórce A2 jest wpisana liczba. Jeśli komórka A2 jest pusta, pojawia się odpowiedni komunikat, w przeciwnym przypadku sprawdza się, czy liczba jest mniejsza czy większa od wylosowanej. W arkuszu wstawiono dwa elementy graficzne do których przypisano odpowiednie makra. Należy program uzupełnić o makro „Czyść” usuwające zawartość komórek A2 i B2, przygotowujące arkusz dla kolejnego gracza. Wygrywa gracz, który w mniejszej liczbie kroków poda wylosowaną liczbę. Napisanie takiej gry nie jest skomplikowane i zadanie takie można wykonać na jednym z pierwszych zajęć z programowania w VBA.

```
Sub Losowanie()  
Randomize  
Cells(1, 10).Value = Int(1000 * Rnd) + 1  
Cells(1, 10).Font.Color = vbWhite  
End Sub  
  
Sub WybranieLiczby()  
Dim dozgadnienia As Integer  
dozgadnienia = Cells(1, 10).Value  
If Cells(2, 1).Value = "" Then  
    Cells(2, 2).Value = "Wpisz liczbę"  
    Exit Sub  
End If  
If Cells(2, 1).Value = dozgadnienia Then  
    Cells(2, 2).Value = "Gratulacje, odgadłeś liczbę!"  
ElseIf Cells(2, 1).Value < dozgadnienia Then  
    Cells(2, 2).Value = "Podaj większą liczbę"  
Else  
    Cells(2, 2).Value = "Podaj mniejszą liczbę"  
End If  
End Sub
```

	A	B
1	Wpisz do komórki A2 szukaną liczbę	
2	155	Podaj większą liczbę
3		
4	Wylosuj liczbę do zgadnięcia	Sprawdź
5		
6		
7		

Rysunek 1. Plansza do gry „Znajdź wylosowaną liczbę”

6. Podsumowanie

Zajęcia fakultatywne z informatyki w szkole podstawowej lub ponadpodstawowej pozwalają uczniom poszerzać wiedzę i umiejętności w zakresie wychodzącym poza podstawę programową, mogą być poświęcone językom programowania innym niż te, które uczniowie poznają na zajęciach. Jak pokazano powyżej, umiejętność programowania w VBA może być wykorzystana nie tylko do automatyzacji prac w arkuszu

kalkulacyjnym, ale także do nauki programowania zarówno poprzez implementację wymaganych przez podstawę programową algorytmów, a także do programowania gier i programów edukacyjnych.

Literatura

1. Jelen B., Syrstad T., Excel 2021 i Microsoft 365. VBA i makra. APN Promise. Warszawa 2022.
2. Kołowska-Gawiejnowicz M., Maćkowiak M., Programowanie funkcyjne w szkole ponadpodstawowej, Informatyka w edukacji, red. Kwiatkowska Anna Beata, Sysło Maciej M., Wydawnictwo Adam Marszałek (2023), str. 148-158.
3. Kołowska-Gawiejnowicz M., Maćkowiak M., Programowanie w Prologu na zajęciach fakultatywnych w szkole ponadpodstawowej, Informatyka w edukacji, red. Kwiatkowska Anna Beata, Sysło Maciej M., Wydawnictwo Adam Marszałek (2025), str. 114-126.
4. Kuciński K., Visual Basic dla Excela w przykładach, Wydawnictwo Witanet, 2015.
5. Walkenbach J., Programowanie w Excel 2000. Visual Basic, Wydawnictwo RM, Warszawa 2000.
6. Zborowska W., Programowanie w VBA na szkolnych zajęciach fakultatywnych, praca licencjacka, WMI UAM, Poznań 2024.
7. Rozporządzenie Ministra Edukacji z dnia 28 czerwca 2024 r. zmieniające rozporządzenie w sprawie podstawy programowej kształcenia ogólnego dla liceum ogólnokształcącego, technikum oraz branżowej szkoły II stopnia, <https://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20240001019> (dostęp: 2.06.2026).