Microsoft

Tracy Syrstad
Bill Jelen

# Microsoft Excel VBA i makra

## Przewodnik po wydajnej automatyzacji

Przekład: Marek Włodarz, Leszek Biolik

APN Promise, Warszawa 2026

**Microsoft Excel VBA i makra: Przewodnik po wydajnej automatyzacji**

Authorized translation from the English language edition, entitled: Microsoft Excel VBA and Macros: Your guide to efficient automation, ISBN: 978-0-13-541023-3, by Tracy Syrstad and Bill Jelen, published by Pearson Education, Inc, publishing as Microsoft Press, a Division of Microsoft Corporation.



Polish language edition published by APN PROMISE S.A., Copyright © 2026

Autoryzowany przekład z wydania w języku angielskim, zatytułowanego: Microsoft Excel VBA and Macros: Your guide to efficient automation, ISBN: 978-0-13-541023-3, by Tracy Syrstad and Bill Jelen, opublikowanego przez Pearson Education, Inc, publikującego jako Microsoft Press, oddział Microsoft Corporation.



APN PROMISE SA, ul. Domaniewska 44a, 02-672 Warszawa
tel. +48 22 35 51 600
e-mail: wydawnictwo@promise.pl



ISBN: 978-83-7541-603-9 (druk), 978-83-7541-607-7 (ebook)

Przekład: Leszek Biolik, Marek Włodarz
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

*Dla Tracy Syrstad. Dziękuję, że od 20 lat jesteś tak świetną współautorką!*

*– Bill Jelen*

*Moim klientom, którzy sprawili, że ta książka była możliwa – Marlee Jo J., Dale W., Eddie G., i wszystkim innym przez te wszystkie lata – dziękuję za ambitne projekty, celne pytania i nieoczekiwane zwroty akcji. Nie tylko utrzymywaliście mnie na nogach – sprawiliście, że stawałam się lepsza.*

*– Tracy Syrstad*

# Spis treści

# Podziękowania

Mówi się, że potrzeba całej armii ludzi, aby celebrytka mogła wkroczyć na czerwony dywan – stylistów, trenerów i całego mnóstwa innej magii, która dzieje się za kulisami. Pisanie książki nie jest aż tak bardzo odmienne. Miałam swój własny zespół dbający o to, aby szwów nie było widać.

Loretta: Ostoja spokoju wśród burzy, strażniczka ciągłości i osoba, która dała radę pokierować tym wszystkim, ani razu nie podnosząc głosu – a przynajmniej nie tam, gdzie mogłabym to usłyszeć.

Bill Jelen: Dziękuję ci za otwarcie drzwi, pokazanie mi drogi i podróżowanie wraz ze mną. To ty sprawiłeś, że w ogóle stało się to możliwe.

Podziękowania należą się też wielu innym (kolejność chaotyczna i przypadkowa). Robert L. dostarczał mi energii (w postaci ciastek); Chris H. zapewnił scenę, na której mogłam podzielić się moimi trikami w VBA; Geoff W. podrzucał mi łamigłówki, które wymagały prawdziwie nieszablonowanego myślenia; no i Ash – zasadniczo klient, ale w rzeczywistości współpracownik – który pomógł myśleć wersjami, a nie tylko znajdować odpowiedzi. Twoja jasność za każdym razem przebijała się przez mgłę. Konsulting pozwolił mi poznać ludzi z każdej branży i ta różnorodność – dziwne błędy, błyskotliwe poprawki, nieoczekiwane pytania – sprawia, że ta praca sprawia tyle radości.

Icarus, MeltedCake i Chilly: Przypominaliście mi, że „It wasn't a phase, Mom" może być sposobem na życie. Dzięki za chaos, lojalność i perfekcyjne zaplanowane rozpraszanie.

Jareth, Luna, Sam i Anita: Wasze codzienne wygłupy warte są karmienia o 2 nad ranem i szaleństw.

No i John: Dziękuję, że jesteś moim oparciem w burzliwych czasach, odkąd porzuciłam korporacyjną harówę i zaczęłam budować własną, dziwną i cudowną ścieżkę.

*– Tracy*

Loretta Yates z Pearsona jest wspaniałym redaktorem prowadzącym. Dziękuję Rickowi Kughenowi za przeprowadzenie tej książki przez meandry produkcji.

Wiele, jeśli nie większość z tego, czego nauczyłem się o programowaniu VBA, zawdzięczam wspaniałej społeczności forum MrExcel.com. VoG, Richard Schollar i Jon von der Heyden szczególnie zasługują na wyróżnienie za ich posty, które wniosły wiele pomysłów wykorzystanych w tej książce. Dziękuję Pam Gensel za pierwszą lekcję z makr Excela. Mala Singh nauczyła mnie tworzenia wykresów VBA. Suat Özgür zadbał, abym był na bieżąco z nowymi trendami w VBA i wniósł wiele pomysłów do rozdziału 18.

Moja rodzina była niezwykle pomocna i cierpliwa w tym czasie. Dziękuję, Mary Ellen Jelen.

*– Bill*

# O autorach

**Bill Jelen**, Excel MVP i gospodarz witryny MrExcel.com, posługuje się arkuszami kalkulacyjnymi od roku 1985, zaś witrynę MrExcel.com utworzył w roku 1998. Jest regularnym gościem wideobloga *Call for Help with Leo Laporte* i wyprodukował ponad 2500 odcinków swojego codziennego podcastu wideo, *Learn Excel from MrExcel*. Jest autorem 71 książek na temat programu Microsoft Excel i comiesięcznego felietonu dla *Strategic Finance*. Przed utworzeniem witryny MrExcel.com Bill Jelen spędził 12 lat na froncie, pracując w działach analiz finansowych, marketingu, księgowości i zarządzania publicznej spółki o kapitale przekraczającym 500 milionów dolarów. Mieszka w Merritt Island na Florydzie ze swą żoną Mary Ellen.

**Tracy Syrstad** jest programistką i projektantką rozwiązań dla Microsoft Excel oraz autorką 11 książek o tej tematyce. Zajmuje się wsparciem w zakresie Microsoft Office od roku 1997, gdy odkryła istnienie publicznych forum online, w których każdy mógł zadać pytanie lub udzielić odpowiedzi. Tracy odkryła, że uczenie innych sprawia jej przyjemność i gdy zaczęła pracować jako projektant, połączyła radość uczenia z codzienną pracą.

# Wprowadzenie

**W tym wprowadzeniu:**

- Dowiemy się, co znajdziemy w tej książce
- Zastanowimy się nad przyszłością VBA i wersji Excela dla Windows
- Poznamy elementy specjalne i konwencje typograficzne stosowane w książce
- Dowiemy się, gdzie znaleźć pliki kodu towarzyszące książce

Kiedy okazało się, że czas realizowania żądań przez działy IT ciągle się wydłuża, użytkownicy programu Excel odkryli, że używając języka makr *Visual Basic for Applications* (VBA) mogą samodzielnie tworzyć raporty potrzebne do prowadzenia przedsiębiorstwa. VBA umożliwia uzyskanie niewiarygodnej efektywności w codziennym użytkowaniu programu Excel. VBA pomaga nam znaleźć sposób na zaimportowanie danych i wygenerowanie raportów w programie Excel, dzięki czemu nie musimy czekać, by w tym zadaniu pomógł nam dział IT.

## Czy TypeScript jest zagrożeniem dla VBA?

Pierwsze pytania, jakie zapewne stawia sobie każdy Czytelnik, to: „Czy powinienem poświęcić czas na naukę VBA? Jak długo Microsoft będzie wspierać VBA? Czy język TypeScript wprowadzony dla wydania Excel Online zastąpi VBA?"

Odpowiedź brzmi – inwestycja w VBA będzie służyć nam dobrze co najmniej do roku 2049.

Ostatnia zmiana języka makr – z XLM na VBA – nastąpiła w roku 1993. Tym niemniej, XLM nadal jest obsługiwany w Excelu. Mamy do czynienia z sytuacją, gdy VBA jest pod każdym względem *lepszy* niż XLM, ale XLM nadal jest wspierany, ponad 30 lat później. Jeśli Microsoft kiedykolwiek zdecyduje się na przejście z VBA na TypeScript, możemy oczekiwać, że wsparcie dla VBA będzie kontynuowane w wersjach Excela dla Windows i Maca przez kolejne 30 lat.

W dzisiejszym uniwersum Excela mamy wersje działające w systemach Windows, w MacOS, na telefonach komórkowych systemu Android i iOS, a także w nowoczesnych przeglądarkach z wykorzystaniem Excel Online. W moim świecie 99% użycia to komputer z systemem Windows. Zapewne w 1% przypadków używam Excela Online. Jeśli jednak ktoś pracuje w środowisku mobilnym, w którym używa Excela w przeglądarce, funkcje UDF w języku TypeScript mogą być odpowiednie.

Jako wprowadzenie do funkcji UDF TypeScript dla Excela można przeczytać książkę Suata M. Özgüra *Excel Custom Functions Straight to the Point* (ISBN 978-1-61547-259-8).

Niemniej jednak wydajność TypeScript jest nadal fatalna. Jeśli nie potrzebujemy makr, które da się uruchamiać w Excel Online, wersja VBA naszych makr będzie działać (co najmniej) osiem razy szybciej, niż wersja TypeScript. Dla tych, którzy zamierzają uruchamiać Excela wyłącznie na platformach Mac lub Windows, VBA będzie językiem wyboru co najmniej przez następną dekadę.

Zagrożeniem dla Excel VBA są natomiast nowe narzędzia Excel Power Query, które można znaleźć w sekcji **Pobieranie i przekształcanie danych** (Get & Transform) karty **Dane** programu Excel dla Windows. Jeśli ktoś tworzy makra w celu czyszczenia importowanych danych, powinien rozważyć sprzątanie ich jednorazowo za pomocą Power Query, a następnie odświeżać zapytanie każdego dnia. Bill ma już mnóstwo opracowanych przebiegów Power Query, które wcześniej wymagały VBA. Jako wprowadzenie do Power Query, warto zapoznać się z książką *Master Your Data with Excel and Power BI: Leveraging Power Query to Get & Transform Your Task Flow* autorstwa Kena Pulsa i Miguela Escobara (ISBN 978-1-61547-058-7).

## Czy Python jest zagrożeniem dla VBA?

Implementacja Pythona w Excelu nie ma na celu automatyzacji żmudnych, powtarzalnych działań, jak VBA. Nie kontroluje skoroszytu, nie odpowiada na zdarzenia ani nie obsługuje zadań formatowania lub operacji plikowych. Jest za to potężną alternatywą dla innych narzędzi analizowania danych w arkuszu – idealnym przy zaawansowanych obliczeniach, przekształcaniu danych za pomocą biblioteki pandas czy tworzeniu niestandardowych wizualizacji. Powinniśmy myśleć o nim jako silniku formuł nowej generacji, a nie narzędzi do makr.

Python w Excel działa jako specjalny rodzaj formuły, wykorzystującej funkcję `=PY()`. Kod Pythona wpisujemy wprost w komórce, zaś Excel przesyła ten kod do zabezpieczonego środowiska chmurowego (piaskownicy) w celu wykonania. Wyniki tych obliczeń są zwracane do arkusza.

Funkcja wspiera takie biblioteki, jak pandas, matplotlib, NumPy i seaborn. Sprawia to, że jest to świetny sposób zaawansowanych analiz, ale Python nie może modyfikować struktury arkusza ani wywoływać innych działań. Na razie przynajmniej oznacza to, że VBA nadal jest podstawowym wyborem dla automatyzacji zadań, kontrolowania zachowania Excela i budowania interaktywnych narzędzi.

# Co zawiera niniejsza książka?

Zakup tej książki to dobra decyzja – pozwoli skrócić i ułatwić proces opanowania umiejętności pozwalających samemu napisać makra VBA, a w konsekwencji zlikwidować obciążenia związane z ręcznym generowaniem raportów.

## Skrócenie procesu przyswajania wiedzy

We Wprowadzeniu przedstawiono analizę przypadku, która ilustruje możliwości makr. W rozdziale 1 „Zwiększanie możliwości programu Excel za pomocą języka VBA" znajdziemy omówienie narzędzi i potwierdzenie znanego faktu: rejestrator makr nie działa stabilnie. Rozdział 2 „Skoro to BASIC, dlaczego nie wygląda znajomo?" ułatwia zrozumienie szalonej składni języka VBA. Rozdział 3 „Odwoływanie się do zakresów, nazw i tabel" wyjaśnia, jak sprawnie pracować z zakresami, komórkami, zdefiniowanymi nazwami i tabelami.

Rozdział 4, „Przygotowywanie podstaw za pomocą zmiennych i struktur" kładzie podwaliny pod pisanie elastycznego, łatwego w utrzymaniu kodu VBA, wprowadzając kluczowe koncepcje procedur, zmiennych i referencji do obiektów oraz narzędzia strukturyzacji kodu, takich jak bloki `With` i dyrektywy kompilatora. Te bloki konstrukcyjne pomogą zrozumieć – a ostatecznie opanować – kod, który będziemy napotykać w tej książce.

W rozdziale 5 „Pętle i sterowanie przepływem" omówiono możliwości mechanizmu pętli w języku VBA. Analiza przypadku w tym rozdziale prezentuje proces tworzenia programu generowania raportu dla działu, a następnie „opakowanie" go procedurą, która za pomocą pętli generuje 46 raportów.

W rozdziale 6 „Formuły w stylu R1C1", rzecz jasna, opisano działanie takich formuł. Rozdział 7 „Programowanie zdarzeń" pokazuje, jak interakcja z użytkownikiem i automatyczne funkcjonalności Excela, takie jak obliczenia, mogą zostać wykorzystane do wyzwalania procedur. Rozdział 8 „Tablice" i rozdział 9 „Tworzenie niestandardowych obiektów i kolekcji" to odpowiednio omówienie tablic, niestandardowych obiektów i kolekcji, doskonałych narzędzi do porządkowania danych. W rozdziale 10 „Formularze użytkownika – wprowadzenie" opisano niestandardowe okna dialogowe, których można używać do zbierania informacji od użytkowników za pomocą programu Excel.

## Potęga Excel VBA

Rozdział 11 „Analiza danych za pomocą funkcji Filtr zaawansowany" i rozdział 12 „Wykorzystywanie VBA do tworzenia tabel przestawnych" to dokładne przedstawienie narzędzi Filtr, Filtr zaawansowany oraz tabeli przestawnych. Narzędzia automatyzacji raportów intensywnie wykorzystują te koncepcje. W rozdziale 13 „Siła programu

Excel" i w rozdziale 14 „Funkcje definiowane przez użytkownika" omówiono dziesiątki przykładów kodu opracowanego w celu zaprezentowania możliwości VBA i funkcji niestandardowych. Nowy podrozdział „Przejście na wyższy poziom: Prawdziwe projekty, prawdziwe problemy, prawdziwe rozwiązania" rozdziału 13 przedstawia różne problemy napotykane w prawdziwych projektach i sposoby ich rozwiązywania.

W rozdziałach od 15 „Tworzenie wykresów" do 20 „Automatyzowanie programu Word" zapoznajemy się z procesami tworzenia wykresów, wizualizacji danych, generowania zapytańdo stron sieci Web i automatyzowania działań w programie Word.

## Materiały techniczne potrzebne do tworzenia aplikacji

W rozdziale 21 „Wykorzystanie programu Access w celu zapewnienia dostępu do danych wielu użytkownikom" omówiono obsługę odczytywania i zapisywania baz danych programu Access i SQL Server. Metody wykorzystywania baz danych Access umożliwiają tworzenie aplikacji z funkcjami obsługi wielu użytkowników, które są dostępne w programie Access, przy jednoczesnym zachowaniu przyjaznego interfejsu użytkownika programu Excel.

W rozdziale 22 „Zaawansowane techniki formularzy użytkownika" rozwinięta została tematyka stosowania formularzy użytkowników. Rozdział 23 „Interfejs programowania aplikacji (API)" zapoznaje nas z niektórymi sposobami realizacji zadań za pomocą interfejsu Windows API. W rozdziałach od 24 „Obsługa błędów" do 26 „Tworzenie dodatków" omówiono obsługę błędów oraz działanie niestandardowych menu i dodatków. Rozdział 27 „Wprowadzenie do tworzenia dodatków pakietu Office" to krótkie wprowadzenie w proces konstruowania własnych aplikacji TypeScript w programie Excel.

## Czy książka ta uczy programu Excel?

Niezupełnie. Zakładamy, że Czytelnicy tej książki już używają Excela, a nawet osiągnęli w nim pewien poziom biegłości i są gotowi, aby pójść dalej za pomocą VBA. Choć nie zamierzamy pokazywać każdej możliwej funkcji, zwrócimy uwagę na prawdziwie potężne narzędzia, takie jak tabele przestawne i Power Query, gdy będzie to istotne dla kodu. Jeśli ktoś nie używał ich wcześniej, będzie wiedział, co warto bliżej poznać.

Bill regularnie prowadzi seminaria Power Excel dla księgowych. Są to podstawowi użytkownicy programu Excel, którzy poświęcają 30 do 40 godzin tygodniowo na pracę z tym programem. Na każdym seminarium pojawiają się dwie kwestie. Pierwsza kwestia to mocne zaskoczenie co najmniej połowy uczestników, jak szybko można zrealizować zadanie za pomocą poszczególnych funkcji, na przykład automatycznego podsumowania czy tabel przestawnych. Druga kwestia to fakt, że zawsze ktoś z uczestników „przebija" Billa. Przykładowo, ktoś zadaje pytanie, odpowiadam na nie, a inna osoba w drugim rzędzie podnosi rękę i daje lepszą odpowiedź.

Wniosek? Sporo wiemy na temat programu Excel. Jednakże oceniam, że w dowolnym rozdziale około 60% osób nie stosowało nigdy wcześniej tabel przestawnych, a jeszcze więcej używało funkcji filtru „10 pierwszych" w tabeli przestawnej. Mając to na uwadze, zdecydowałem, że zanim pokażę, jak zautomatyzować cokolwiek w VBA, krótko omawiam wykonanie tego samego zadania w interfejsie programu Excel. Niniejsza książka nie uczy nas, jak tworzyć tabele przestawne, ale jedynie zwraca uwagę na tematykę, z którą warto zapoznać się dokładniej.

## Studium przypadku: Miesięczne raporty księgowe

To jest prawdziwa historia. Valerie jest analitykiem biznesowym w dziale księgowości średniej wielkości korporacji. W jej firmie niedawno zainstalowano system ERP (*enterprise resource planning*), który przekroczył budżet o 16 milionów dolarów. Pod koniec projektu okazało się, że zabrakło funduszy w budżecie IT na generowanie miesięcznych raportów wykorzystywanych w firmie do podsumowywania działań każdego działu.

Jednak Valerie była już bliska podjęcia decyzji o tworzeniu raportów własnymi siłami. Wiedziała, że w tym celu konieczne będzie wyeksportowanie danych księgi głównej z systemu ERP do pliku tekstowego o wartościach rozdzielanych przecinkami. Za pomocą Excela, Valerie potrafiła zaimportować dane księgi głównej z systemu ERP do arkusza.

Tworzenie raportu nie było proste. Podobnie jak w wielu innych firmach, w danych pojawiały się wyjątki. Valerie wiedziała, że niektóre konta w jednym ze źródeł kosztów powinny być przeklasyfikowane jako wydatki oraz że inne konta trzeba całkowicie usunąć z raportu. Pracując uważnie w programie Excel, Valerie wykonała te dostosowania. Utworzyła tabelę przestawną, by wygenerować pierwszą część podsumowania raportu. Wycięła wyniki tabeli przestawnej i wkleiła je do pustego arkusza. Następnie utworzyła nowy raport tabeli przestawnej dla drugiej części podsumowania. Po około trzech godzinach miała zaimportowane dane, utworzone i poukładane do podsumowania pięć tabel przestawnych oraz starannie sformatowany raport.

### Zostać bohaterem

Valerie zaniosła raport menedżerowi, który właśnie usłyszał od działu IT, że generowanie tego zawiłego raportu mogą zabrać dwa, trzy miesiące. Po utworzeniu raportu w programie Excel, Valerie natychmiast stała się bohaterem dnia. W trzy godziny zrobiła coś niemożliwego do wykonania. Valerie była w siódmym niebie, kiedy usłyszała zasłużone „zuch dziewczyna".

### Kolejne zachwyty

Następnego dnia menedżer Valerie uczestniczył w comiesięcznym spotkaniu działów. Kiedy inni menedżerowie działów zaczęli narzekać, że nie mogą uzyskać raportów z systemu ERP, menedżer Valerie położył swój raport na stół. Pozostali byli bardzo zdumieni, jak można było wygenerować ten raport? Każdy był zadowolony, że udało się komuś „złamać kod". Szef firmy poprosił menedżera Valerie, czy nie mogłaby przygotowywać raportu dla każdego działu.

### Radość przemienia się w strach

Łatwo domyślić się, co się stało. W tej konkretnej firmie było 46 działów. Oznacza to 46 jednostronicowych podsumowań, które trzeba generować co miesiąc. Każdy raport wymaga importowania danych z systemu ERP, usunięcia pewnych kont, utworzenia pięciu tabel przestawnych i sformatowania raportu w kolorze. Utworzenie pierwszego raportu zajęło Valerie trzy godziny, a po nabyciu wprawy, można oszacować, że wygenerowanie 46 raportów zajmie jej około 40. Nawet jeśli Valerie jeszcze bardziej skróci ten czas, i tak jest to horror. Valerie miała wcześniej inne zadania do wykonania, zanim stała się odpowiedzialna za „spędzenie" 40 godzin na generowaniu tych raportów.

### Ratunkiem jest VBA

Valerie znalazła moją firmę, MrExcel Consulting i wyjaśniła, na czym polega problem. W ciągu tygodnia napisałem kilka makr w VBA, które realizowały wszystkie powtarzalne zadania. Na przykład makra importowały dane, usuwały zbędne konta, tworzyły 5 tabel przestawnych i formatowały raporty. Inaczej mówiąc, 40-to godzinny ręczny proces został zredukowany do kliknięcia dwóch przycisków i skrócony do około 4 minut.

Niewątpliwie w każdej firmie znajdzie się ktoś przyzwyczajony do ręcznego wykonywania zadań, które można zautomatyzować za pomocą VBA. Jestem przekonany, że mogę pójść do każdej firmy, w której jest ponad 20 użytkowników programu Excel i znajdę podobnie zdumiewający przypadek, jaki spotkał Valerie.

# Wersje programu Excel

To ósme już wydanie książki *VBA i makra* zostało opracowane do współpracy z programem Excel 365 według stanu dostępnego w lipcu 2025 roku. Poprzednie wydania tej książki uwzględniały kod dla wersji od Excel 97 po Excel 2019 i Excel 365 (2021). W 80% obecny kod jest identyczny z kodem dla poprzednich wersji*.

## Różnice dla użytkowników systemu Mac

Pomimo że program Excel dla systemu Windows i program Excel dla systemu Mac są podobne w kontekście interfejsu, istnieje jednak szereg różnic, jeśli będziemy porównywać środowisko VBA. Rzecz jasna, nic z rozdziału 23, który wykorzystuje interfejs Windows API, nie będzie działać w systemie macOS. Tym niemniej można stwierdzić, że ogólne koncepcje omawiane w tej książce stosują się także do Maców. Ogólną listę różnic w odniesieniu do macOS znaleźć można pod adresem *http://www.mrexcel.com/macvba.html*. VBA Editor dla Maca nie umożliwia projektowania UserForms (rozdział 10). Istnieje także nadal nieusunięty bug, który utrudnia tworzenie makr obsługi zdarzeń(rozdział 7). Excel zgłasza błąd, gdy próbujemy wybierać z list rozwijanych widocznych u góry okna Code. Trzeba najpierw skopiować i wkleić pustą procedurę zdarzenia – wówczas te listy rozwijane będą działać.

# Elementy specjalne i konwencje typograficzne

W książce używane są następujące konwencje typograficzne:

- *Kursywa* – wskazuje nowe pojęcie podczas jego definiowania, specjalne wyróżnienie, obce słowa czy frazy oraz adresy internetowe.

* Ze względu na fakt, że w Polsce zazwyczaj mamy do czynienia z polską wersją programu Excel, nazwy funkcji, a także poleceń i elementów interfejsu podawane są w języku polskim. Na końcu książki zamieściliśmy tabele zawierające nazwy funkcji występujące w polskiej wersji programu oraz ich angielskie odpowiedniki. Nazwy funkcji użytych w formułach są automatycznie tłumaczone, gdy skoroszyt utworzony w wersji angielskiej zostanie otwarty w wersji polskiej bądź odwrotnie. Nie dotyczy to jednak statycznego tekstu (komentarzy/opisów) zawartego w plikach przykładowych, które pozostają w wersji oryginalnej, co widać również na zawartych w książce zrzutach ekranowych. Automatycznej konwersji podlegają również formaty dat oraz znak dziesiętny, którym w wersji polskiej jest przecinek, a w angielskiej kropka. Trzeba też pamiętać, że w formułach w wersji polskiej zmieniany jest znak separatora – w wersji angielskiej jest to przecinek, zaś w polskiej średnik.
Powyższe uwagi *nie dotyczą* jednak kodu w języku VBA – tu nazwy funkcji, separatory itp. pozostają bez zmian, gdyż wspomniana automatyczna konwersja dotyczy jedynie *wyświetlania* w interfejsie użytkownika, a nie rzeczywistej zawartości skoroszytu programu Excel (wszystkie przypisy pochodzą od redakcji wydania polskiego).

- `Czcionka o stałej szerokości` – Wskazuje część kodu VBA, na przykład nazwę obiektu czy metody.
- **Czcionka wytłuszczona bezszeryfowa** – Wskazuje nazwy elementów interfejsu użytkownika oraz informacje wprowadzane przez użytkownika.

Oprócz tych konwencji typograficznych używanych jest też kilka elementów specjalnych. W każdym rozdziale jest co najmniej jedno *Studium przypadku*, które opisuje rzeczywiste rozwiązania typowych problemów. Analizy przypadków pokazują także praktyczne zastosowania tematyki omawianej w rozdziale.

Oprócz wspomnianych Studiów przypadku zobaczymy również ramki Wyższy poziom, Uwaga, Wskazówka i Ostrzeżenie:

**Wyższy poziom!** Te notatki oferują pogłębione spostrzeżenia lub sprytniejsze techniki wykraczające poza podstawy. To opcjonalne, dodatkowe obszary do eksploracji – doskonałe, jeśli ktoś jest gotowy na rozwinięcie swoich umiejętności nieco bardziej, niż zakłada bieżący rozdział.

**Uwaga** Uwagi udostępniają dodatkowe, przydatne informacje spoza głównego wątku omawianych kwestii.

**Wskazówka** Wskazówki opisują szybkie inne sposoby rozwiązywania omawianego problemu, które oszczędzają czas i umożliwiają bardziej efektywną pracę.

**Ostrzeżenie** Ostrzeżenia wskazują na potencjalne problemy i pułapki, z którymi możemy się zetknąć. Warto zapoznać się z tymi informacjami – mogą oszczędzić nam wielu godzin pracy i zdenerwowania.

# Pliki kodu

W podziękowaniu za zakup tej książki, dołączyliśmy zestaw blisko 50 arkuszy programu Excel, które ilustrują koncepcje omawiane w książce. Ten zestaw plików obejmuje cały kod omówiony w książce, dane przykładowe oraz dodatkowe uwagi od autorów. Pliki kodu można pobrać ze strony dostępnej pod adresem

*MicrosoftPressStore.com/XLVBAAuto/downloads*

# Errata i aktualizacje

Dołożyliśmy wszelkich starań, aby zapewnić dokładność tej książki i dołączonych do niej materiałów dodatkowych. Aktualizacje – w formie listy zgłoszonych błędów i poprawek – są dostępne pod adresem *MicrosoftPressStore.com/XLVBAAuto/errata*

Jeśli ktoś zauważy błąd, którego nie ma na tej liście, można go zgłosić, korzystając z tej samej strony.

Dodatkowe informacje i wsparcie są dostępne pod adresem *MicrosoftPressStore.com/Support*

Należy zauważyć, że wsparcie dla oprogramowania i sprzętu oferowanego przez firmę Microsoft nie jest dostępne za pośrednictwem wymienionych adresów. Pomoc dotyczącą oprogramowania lub sprzętu firmy Microsoft można uzyskać pod adresem *http://support.microsoft.com*.

ROZDZIAŁ 1

# Zwiększanie możliwości programu Excel za pomocą języka VBA

**W tym rozdziale:**

- Poznamy możliwości programu Excel
- Zrozumiemy dwie główne bariery w nauce języka VBA
- Poznamy swoje narzędzia: karta Deweloper
- Dowiemy się, które typy plików umożliwiają tworzenie makr
- Zapoznamy się z zabezpieczeniami makr
- Przejrzymy opcje rejestrowania, przechowywania i uruchamiania makr
- Poznamy Edytor VB
- Zrozumiemy wady rejestratora makr

**Visual Basic for Applications** (VBA) w połączeniu z Microsoft Excel jest zapewne najbardziej wydajnym spośród dostępnych narzędzi. Choć jest dostępny w komputerach 850 milionów użytkowników Microsoft Office, większość z nich nigdy nawet nie spróbowała wykorzystać mocy VBA w Excelu. Przy użyciu VBA można przyśpieszyć niemal każde zadanie wykonywane w Excelu. Jeśli na przykład regularnie tworzysz w Excelu serie wykresów prezentujących miesięczne wyniki, możesz sprawić, aby VBA wykonał to zadanie w ciągu kilku sekund.

## Początkowe przeszkody

Istnieją dwie przeszkody utrudniające opanowanie programowania w VBA. Po pierwsze, rejestrator makr w Excelu nie jest daleki od doskonałości i nie generuje działającego kodu, który możnaby wykorzystywać jako model. Po drugie, dla wielu osób, które wcześniej poznały taki język programowania jak BASIC, składnia języka VBA jest okropnie frustrująca.

## Rejestrator makr nie działa!

Firma Microsoft zaczęła dominować na rynku arkuszy kalkulacyjnych w połowie lat 90-tych. Wprawdzie odniosła znaczący sukces, tworząc wydajny program arkusza kalkulacyjnego, na który łatwo mogliby przejść użytkownicy programu Lotus 1-2-3, jednak w odniesieniu do makr było zupełnie inaczej. Niemal dla każdego, kto sprawnie tworzył makra 1-2-3, próby rejestrowania makr w Excelu zazwyczaj kończyły się niepowodzeniem. Choć język programowania VBA ma nieporównywalnie większe możliwości, niż język makr programu Lotus 1-2-3, podstawową wadą było niewłaściwe działanie rejestratora makr przy domyślnych ustawieniach.

W programie Lotus 1-2-3 możemy zarejestrować makro jednego dnia, odtworzyć je następnego i zazwyczaj nie będziemy mieli problemów z jego działaniem. Przy próbie wykonania tych samych czynności w Excelu makro może działać dzisiaj, ale jutro już nie. W 1995 roku próba zarejestrowania mojego pierwszego makra w Excelu przysporzyła mi wielu frustracji. W tej książce pokażę trzy reguły, które pozwalają najbardziej efektywnie wykorzystywać rejestrator makr.

## Nikt w zespole programu Excel nie poświęca wiele uwagi rejestratorowi makr

Kiedy w firmie Microsoft dodawane są nowe funkcje w programie Excel, poszczególni menedżerowie projektu danej funkcji zakładają, że rejestrator makr podczas wykonywania polecenia zarejestruje odpowiednią sekwencję. W poprzedniej dekadzie, rejestrowany kod *mógł* działać w niektórych sytuacjach, ale zwykle nie działał poprawnie *we wszystkich* sytuacjach. Gdyby ktoś w firmie Microsoft był bardziej zainteresowany stworzeniem przydatnego rejestratora makr, rejestrowany kod mógłby być nieco bardziej ogólny, niż jest obecnie.

Spodziewaliśmy się, że makro można nagrać dowolną z pięciu dostępnych metod, a zarejestrowany kod będzie działał. Niestety obecnie, jeśli chcemy zastosować rejestrator makr, musimy często próbować rejestrowania makra siedmioma różnymi metodami, aż znajdziemy taki zestaw kroków, które rejestrują stabilnie działający kod.

## Visual Basic nie przypomina języka BASIC

Kiedy pierwszy raz ujrzałam kod wygenerowany przez rejestrator makr, nie przypominał niczego, z czym się spotkałam wcześniej. Nazywano to „Visual Basic" (VB). Przez lata miałam okazję poznać kilka innych języków programowania, ale ten dziwacznie wyglądający język był zupełnie nieintuicyjny i zdecydownie nie przypominał języka BASIC, którego uczyłam się jeszcze w szkole średniej.

Dla Billa, który miał zdecydowanie większe doświadczenie w programowaniu ode mnie, było to jeszcze trudniejsze. Już w 1995 roku był uważany w swoim biurze za eksperta od arkuszy kalkulacyjnych. W tym właśnie czasie nakazano wszystkim

w firmie przejście z Lotusa 1-2-3 do Excela, co oznaczało dla niego konieczność zmagania się z rejestratorem makr, który nie działał, i z językiem, którego nie rozumiał. Trudno to nazwać korzystnym splotem wydarzeń

Jednym z założeń, które przyjęliśmy podczas pisania tej książki, jest to, że Czytelnik jest sprawnym użytkownikiem arkuszy kalkulacyjnych oraz że prawdopodobnie umie więcej, niż 90% osób w jego firmie. Ponadto zakładamy, że nawet jeśli nie jest programist(k)ą, zapewne zetknął się z językiem BASIC gdzieś na swojej drodze edukacyjnej. Jednakże znajomość tego języka nie jest wymaganiem – w istocie jest przeszkodą w osiągnięciu celu, jakim jest sprawne programowanie w języku VBA. Istnieje również spora szansa, że Czytelnik próbował rejestrować makro w Excelu i podobna szansa, że nie był zadowolony z uzyskanych wyników.

**Uwaga** Jeśli znasz już podstawy programowania – albo nawet jesteś doświadczonym użytkownikiem VBA – nie pozwól, aby cię zwiodły te początkowe rozdziały. Choć zaczynamy od samych podstaw, szybko przejdziemy do bardziej zaawansowanych obszarów: czystszego kodu, bardziej inteligentnych struktur i rozwiązań zaprojektowanych do praktycznego zastosowania. Niezależnie od tego, czy uczysz się od początku, czy też przychodzisz ze świata innego języka programowania, znajdziesz tu mnóstwo treści, które pozwolą ci rozszerzyć swoje umiejętności i usprawnić pracę.

## Dobra wiadomość: poznawanie języka VBA nie jest trudne

Jeśli nawet próba zastosowania rejestratora makr okazała się frustrująca, można to uznać jedynie za niewielką przeszkodę na drodze do pisania złożonych programów w programie Excel. Z tej książki dowiemy się nie tylko, dlaczego nie działa rejestrator makr, ale także, jak zmodyfikować zarejestrowany kod, by stał się użyteczny.

## Dobra wiadomość: Excel plus VBA są warte wkładanego wysiłku

Zapewne brak możliwości skutecznego rejestrowania makr w Excelu spowodował, że wiele osób poczuło się rozczarowanych firmą Microsoft. Trzeba jednak podkreślić, że możliwości języka VBA w programie Excel są ogromne. Niemal wszystko, co można wykonać za pomocą interfejsu programu, można niezwykle szybko osiągnąć używając języka VBA. Wyjątki, takie jak wstawienie opisu dla funkcji LAMBDA, są naprawdę rzadkie. Jeśli ktoś codziennie czy co tydzień ręcznie tworzy takie same raporty, język VBA w programie Excel znacznie uprości wykonywanie takich zadań

Autorzy tej książki zapewniają doradztwo w zakresie programu Excel od ponad 20 lat. W ramach tej pracy zautomatyzowaliśmy tworzenie raportów dla setek klientów. Te historie zazwyczaj są do siebie podobne: dział IT ma wielomiesięczną listę

zaległości. Ktoś z działu księgowego lub konstrukcyjnego odkrywa, że można zaimportować pewne dane do programu Excel i utworzyć raporty potrzebne do działania przedsiębiorstwa. „Odkrycie" to pozwala nie czekać już przez wiele miesięcy, aż dział IT napisze program. Problem polega jednak na tym, że po zaimportowaniu danych do programu Excel i zdobyciu uznania u przełożonego za utworzenie raportu, najprawdopodobniej czekać będzie nas dodatkowa i uciążliwa praca związana z tygodniowym czy comiesięcznym generowaniem tego raportu. To zaś może okazać się bardzo żmudne.

I ponownie, doskonałą w tej sytuacji wiadomością jest to, że wystarczy poświęcić kilka godzin na programowanie w języku VBA, by zautomatyzować proces tworzenia raportu i sprowadzić go do kilku kliknięć myszą. Satysfakcja jest ogromna, tak więc pozostańcie ze mną – wyjaśnię kilka podstawowych zasad.

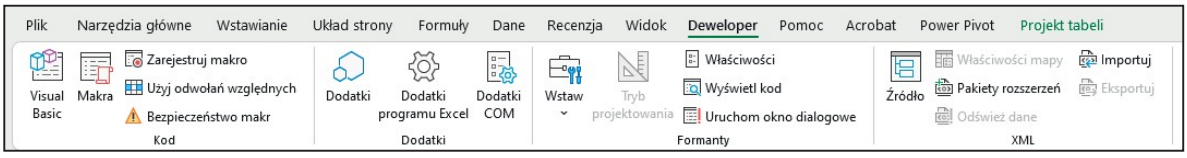
W tym rozdziale wyjaśnimy, dlaczego rejestrator makr nie działa. Przeanalizujemy także przykład zarejestrowanego kodu i pokażemy, dlaczego jednego dnia kod działa, a następnego już nie. W rozdziale znajdują się fragmenty kodu. Zdaję sobie sprawę, że prezentowany w rozdziale kod może nie być (jeszcze) zrozumiały, ale nie trzeba się tym przejmować. Celem tego rozdziału jest przedstawienie podstawowego problemu dotyczącego rejestratora makr. W rozdziale zaprezentowano również podstawowe elementy środowiska programistycznego Visual Basic.

# Poznawanie narzędzi: karta Deweloper

Rozpoczniemy od przeglądu podstawowych narzędzi potrzebnych do korzystania z języka VBA. Domyślnie firma Microsoft ukrywa narzędzia VBA. Aby uzyskać dostęp do karty Deweloper, należy wykonać następujące czynności:

1. Kliknij wstążkę prawym przyciskiem myszy i wybierz polecenie **Dostosuj Wstążkę**.
2. W okienku z prawej strony zaznacz pole wyboru **Deweloper**.
3. Kliknij **OK**, by powrócić do programu Excel.

W programie Excel wyświetlona zostanie karta Deweloper (rysunek 1.1).




**Rysunek 1.1** Karta Deweloper dostępnia interfejs do uruchamiania i rejestrowania makr.

W grupie **Kod** karty **Deweloper** znajdują się ikony używane do rejestrowania i odtwarzania makr VBA:

- **Visual Basic** Otwiera narzędzie Visual Basic Editor.

- **Makra** Kliknięcie ikony powoduje wyświetlenie okna dialogowego Makro, gdzie można uruchomić lub edytować makra wymienione na liście.
- **Zarejestruj makro** Kliknięcie ikony rozpoczyna proces rejestrowania makra.
- **Użyj odwołań względnych** Funkcja ta przełącza pomiędzy trybami rejestrowania względnego i bezwzględnego. W przypadku rejestrowania względnego program Excel rejestruje przemieszczenie w dół o trzy komórki. Dla rejestrowania bezwzględnego program Excel zarejestruje wybranie komórki A4.
- **Bezpieczeństwo makr** Ikona umożliwia dostęp do modułu Centrum zaufania, gdzie można zezwolić na uruchamianie makr lub tego zabronić na tym (lokalnym) komputerze.

W grupie **Dodatki** udostępniono narzędzia zarządzania dodatkami Office, zwykłymi dodatkami i dodatkami COM.

Grupa **Formanty** karty Deweloper zawiera menu **Wstaw**, gdzie znajduje się szereg kontrolek, które można umieszczać na arkuszach. Patrz „Inne sposoby uruchamiania makr" w rozdziale 25. Inne ikony tej grupy umożliwiają pracę z kontrolkami umieszczonymi w arkuszu.

---

**Uwaga** Przycisk **Uruchom okno dialogowe** to starsza funkcja zachowana dla wstecznej kompatybilności, która nie ma praktycznego zastosowania w nowszych wersjach Excela. Oryginalnie była powiązana z niestandardowymi oknami dialogowymi i działaniem starszych formularzy, ale obecnie jest w większości przestarzała i nieaktualna. Większość użytkowników może całkowicie ją zignorować.

---

W grupie **XML** na karcie Deweloper umieszczono narzędzia do importowania i eksportowania dokumentów XML.

# Typy plików zezwalające na makra

Excel obsługuje cztery typy plików skoroszytów. Makra nie mogą być przechowywane w plikach .xlsx, a jak wiadomo, ten rodzaj plików jest typem domyślnym! Musimy używać funkcji **Zapisz jako** dla wszystkich arkuszy zawierających makra, ale można też zmienić domyślny typ pliku używany przez program Excel.

Poniżej wymieniono dostępne typy plików:

- **Skoroszyt programu Excel (.xlsx)** Pliki są przechowywane jako kolekcja obiektów XML, spakowane do pojedynczego pliku. Dzięki temu powstają pliki o znacznie mniejszych rozmiarach, a także możliwa jest edycja lub tworzenie skoroszytów programu Excel w innych aplikacjach (nawet w Notatniku!). Niestety makra nie mogą być przechowywane w plikach z rozszerzeniem .xlsx.

- **Skoroszyt programu Excel z obsługą makr (.xlsm)** Jest to podobny typ pliku do domyślnego .xlsx, ale z włączoną obsługą makr. Podstawowe założenie jest takie, by użytkownik nie musiał obawiać się złośliwych makr, mając plik .xlsx. Jeśli jednak użytkownik widzi plik .xlsm, powinien zwrócić uwagę na to, że do pliku mogą być dołączone makra.
- **Skoroszyt binarny programu Excel (.xlsb)** Jest to binarny format opracowany w celu obsługi tabel zawierających ponad milion wierszy (możliwość wprowadzona w programie Excel 2007). Starsze wersje programu Excel przechowują swoje pliki we własnych formatach binarnych. Pomimo że formaty binarne mogą ładować się szybciej, są mniej odporne na uszkodzenia, a utrata kilku bitów może zniszczyć cały plik. W tym formacie makra są obsługiwane.
- **Skoroszyt programu Excel 97-2003 (.xls)** Format ten tworzy pliki, które mogą być odczytywane przez użytkowników starszych wersji programu Excel. W tym formacie binarnym dopuszczona jest obsługa makr; jeśli jednak zapiszemy plik w tym formacie, utracimy dostęp do komórek spoza zakresu A1:IV65536. Ponadto jeśli otworzymy plik w programie Excel 2003, utracimy dostęp do elementów, korzystających z funkcji wprowadzonych w programie Excel 2007 lub nowszym.

Aby uniknąć konieczności wybierania typu skoroszytu z włączoną obsługą makr za każdym razem, gdy zapisujemy nowy plik, możemy dostosować naszą kopię programu Excel, tak by nowe pliki były zawsze zapisywane w formacie .xlsm. W tym celu:

1. Kliknij menu **Plik** i wybierz menu **Opcje**.
2. W oknie dialogowym **Opcje programu Excel**, w okienku z lewej strony zaznacz kategorię **Zapisywanie**.
3. Wyświetl listę rozwijaną **Zapisz pliki w następującym formacie** i wybierz opcję **Skoroszyt programu Excel z obsługą makr**. Kliknij **OK**.

**Uwaga** Chociaż my (autorzy i wy, Czytelnicy) nie boimy się używać makr, niektórzy użytkownicy się denerwują, gdy widzą plik z rozszerzeniem .xlsm. Format ten może włączyć czerwoną flagę dla użytkowników, którzy mają bardziej restrykcyjne ustawienia lub pracują w środowiskach o podwyższonych zabezpieczeniach.

Jeśli spotkamy kogoś, kto obawia się plików typu .xlsm, warto przypomnieć, że:

- Każdy skoroszyt starszego formatu (.xls) utworzony w ciągu minionych lat mógł mieć makra, choć w rzeczywistości większość ich nie ma.
- Jeśli ktoś chce uniknąć plików z makrami, powinien stosować ustawienia zabezpieczeń, by zapobiec uruchamianiu makr. Osoba taka będzie nadal mogła otwierać plik .xlsm, by uzyskać dostęp do danych zawartych w arkuszu.

Mam nadzieję, że dzięki tym argumentom pokonamy wszystkie obawy dotyczące plików .xlsm i staną się one typem domyślnym.

# Bezpieczeństwo makr

Po tym, jak wykorzystano makra VBA w programie Word jako metodę przesyłania wirusa Melissa, firma Microsoft zmieniła domyślne ustawienia zabezpieczeńtak, by blokować uruchamianie makr. Z tego powodu, zanim rozpoczniemy omawianie sposobów rejestrowania makr, warto przyjrzeć się, jak dostosować ustawienia domyślne.

W Excelu można globalnie zmodyfikować ustawienia zabezpieczeń lub kontrolować ustawienia makr dla wskazanych skoroszytów poprzez przechowywanie ich w zaufanej lokalizacji. Makra zostaną automatycznie włączone dla wszystkich skoroszytów zapisanych w lokalizacji oznaczonej jako zaufana.

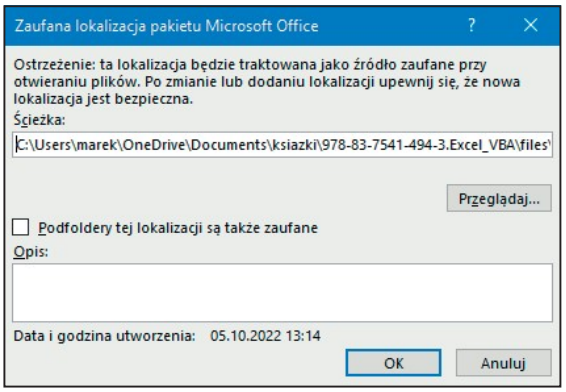
Ustawienia bezpieczeństwa makr znajdziemy, klikając ikonę **Bezpieczeństwo makr** na karcie Deweloper, co spowoduje wyświetlenie kategorii **Ustawienia makr** w oknie **Centrum zaufania**. Okienko nawigacji z lewej strony pozwala uzyskać dostęp do listy **Zaufane lokalizacje**.

## Dodawanie zaufanej lokalizacji

Skoroszyty zawierające makra możemy przechowywać w folderze oznaczonym jako lokalizacja zaufana. Wszystkie skoroszyty zapisane w zaufanym folderze będą miały włączoną obsługę makr. Firma Microsoft zaleca, by zaufane lokalizacje były definiowane na lokalnych dyskach twardych. Zgodnie z ustawieniami domyślnymi nie możemy ufać lokalizacjom na dyskach sieciowych.

Aby zdefiniować zaufaną lokalizację, wykonaj poniższą procedurę:

1. Kliknij przycisk **Bezpieczeństwo makr** na karcie **Deweloper**.
2. W okienku nawigacji z lewej strony okna **Centrum zaufania** kliknij **Zaufane lokalizacje**.
3. Jeśli zaufane lokalizacje mają znajdować się na dyskach sieciowych, zaznacz opcję **Zezwalaj na zaufane lokalizacje w mojej sieci**.
4. Kliknij przycisk **Dodaj nową lokalizację**. Wyświetlone zostaje okno dialogowe **Zaufana lokalizacja pakietu Microsoft Office** (rysunek 1.2).
5. Kliknij przycisk **Przeglądaj**. Program Excel wyświetli okno dialogowe **Przeglądaj**.
6. Przejdź do folderu nadrzędnego dla folderu, który ma stać się lokalizacją zaufaną. Kliknij zaufany folder. Kliknij **OK**. Nazwa folderu pojawi się w polu **Ścieżka**.
7. Aby podfoldery wybranego folderu również były lokalizacjami zaufanymi, zaznacz pole wyboru P**odfoldery tej lokalizacji są także zaufane**.
8. Kliknij **OK**, by dodać folder do listy **Zaufane lokalizacje**.




**Rysunek 1.2** Zarządzanie zaufanymi folderami w oknie Centrum zaufania.

**Ostrzeżenie** Podczas wybierania zaufanych lokalizacji należy zachować ostrożność. Po kliknięciu załącznika wiadomości e-mail, będącego plikiem programu Excel, program Outlook zapisze ten plik w folderze tymczasowym na dysku C. Z tego względu nie będziemy chcieli dodawać całego dysku C:\ i wszystkich jego podfolderów do listy Zaufane lokalizacje.

## Zastosowanie ustawień makr w celu włączenia obsługi makr poza zaufanymi lokalizacjami

Do wszystkich makr zapisanych poza zaufanymi lokalizacjami program Excel stosuje ustawienia makr. Nazwy ustawień Niskie, Średnie, Wysokie i Bardzo wysokie, które znamy z programu Excela 2003, zostały zmienione

Aby uzyskać dostęp do ustawień makr, należy kliknąć polecenie **Bezpieczeństwo makr** na karcie Deweloper. Excel wyświetli kategorię **Ustawienia makr** okna dialogowego **Centrum zaufania**. Należy zaznaczyć drugą opcję: **Wyłącz wszystkie makra i wyświetl powiadomienie**. Poniżej przedstawiono opis każdej opcji:

- **Wyłącz makra języka VBA bez powiadomienia** Ustawienie to blokuje uruchamianie wszystkich makr i jest przeznaczone dla osób, które nigdy nie zamierzają korzystać z makr. Ponieważ ta książka uczy, jak używać makr, nie będziemy używać tej opcji. W przypadku tego ustawienia mogą działać jedynie makra zapisane w zaufanych lokalizacjach.
- **Wyłącz makra języka VBA z powiadomieniem** Istotne słowa w nazwie tej opcji to „z powiadomieniem". Oznacza to, że użytkownikowi wyświetlone zostanie powiadomienie, kiedy otworzy plik zawierający makra, po czym może zdecydować, czy chce włączyć zawartość. Jeśli powiadomienie zostanie zignorowane, makra pozostaną wyłączone. Jest to ustawienie zalecane. W obszarze komunikatów wyświetlane jest powiadomienie informujące o tym, że makra zostały wyłączone. Klikając tę opcję, użytkownik może włączyć zawartość (rysunek 1.3).


**Rysunek 1.3.** Przycisk Włącz zawartość widoczny w przypadku użycia opcji Wyłącz wszystkie makra i wyświetl powiadomienie.

- **Wyłącz makra języka VBA oprócz makr podpisanych cyfrowo** Ustawienie to wymaga użycia narzędzia do tworzenia podpisów cyfrowych od jakiegoś dostawcy zaufania, takiego jak VeriSign. Jest to właściwy wybór dla osób zamierzających sprzedawać dodatki innym użytkownikom, ale jest trochę uciążliwym rozwiązaniem w przypadku pisania makr na własny użytek.
- **Włącz makra języka VBA (niezalecane, może zostać uruchomiony potencjalnie niebezpieczny kod)** Ustawienie to jest podobne do ustawienia **Niskie** w programie Excel 2003. Chociaż ustawienie sprawia najmniej kłopotów, jednak naraża komputer na ataki złośliwych wirusów, takich jak (nie)sławny wirus Melissa. Firma Microsoft nie zaleca używania tego ustawienia.

## Stosowanie opcji Wyłącz makra języka VBA z powiadomieniem

Zalecane jest stosowanie ustawienia **Wyłącz makra języka VBA z powiadomieniem**. Jeśli użyjemy tej opcji, po otwarciu skoroszytu zawierającego makra bezpośrednio nad paskiem formuły wyświetli się ostrzeżenie o zabezpieczeniach. Jeśli spodziewamy się, że ten skoroszyt zawiera makra, należy kliknąć przycisk **Włącz zawartość**. Jeśli nie chcemy włączać makr w otwieranym skoroszycie, możemy zamknąć ostrzeżenie o zabezpieczeniach poprzez kliknięcie ikonki X z prawej strony paska tytułu.

Jeśli zapomnimy włączyć makra i spróbujemy je uruchomić, program Excel poinformuje nas, że nie można uruchomić makra, ponieważ wszystkie zostały wyłączone. W takiej sytuacji najlepiej jest zamknąć skoroszyt i otworzyć go jeszcze raz.

---

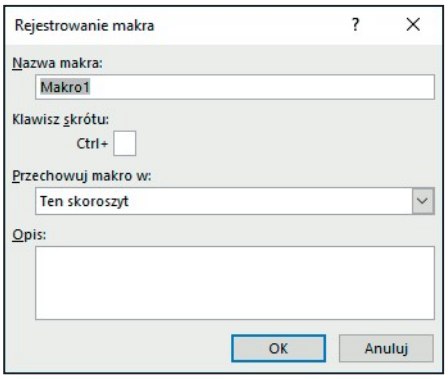
**Ostrzeżenie** Jeśli skoroszyt z włączoną obsługą makr pobierzemy z sieci Web albo dostaniemy go w email, Excel może zablokować plik i wyłączyć makra bez żadnego powiadomienia. Aby tego uniknąć, prawym przyciskiem myszy kliknij pobrany plik, wybierz **Właściwości**, zaznacz pole **Odblokuj** (jeśli jest obecne) i kliknij **OK**.

---

# Rejestrowanie, zapisywanie i uruchamianie makr

Rejestrowanie makr jest przydatne dla osób, które nie mają odpowiedniego doświadczenia, by samodzielnie napisać ich kod. Po zdobyciu potrzebnej wiedzy i doświadczenia coraz rzadziej korzysta się z możliwości rejestrowania makr.

Aby rozpocząć rejestrację makra, na karcie Deweloper wybieramy polecenie **Zarejestruj makro**. Przed rozpoczęciem rejestrowania program Excel wyświetla okno dialogowe **Rejestrowanie makra** (rysunek 1.4).




**Rysunek 1.4.** W oknie dialogowym Rejestrowanie makra, do rejestrowanego makra przypisujemy nazwę i klawisz skrótu

## Wypełnianie okna dialogowego Rejestrowanie makra

W polu **Nazwa makra** wpisujemy jego nazwę. Trzeba pamiętać, by nie używać spacji, przykładowo należy użyć nazwy **Makro1**, a nie **Makro 1**. Ponieważ makr może być bardzo dużo, warto stosować nazwy opisowe, na przykład nazwa FormatowanieRaportu jest znacznie bardziej przydatna niż Makro1.

W drugim polu okna dialogowego Rejestrowanie makra definiujemy klawisz skrótu. Jeśli w tym polu wpiszemy małą literę j, a następnie naciśniemy klawisze Ctrl+J, makro to zostanie uruchomione. Warto pamiętać jednak, że większość skrótów z literami, począwszy od Ctrl+A, a skończywszy na Ctrl+Z (za wyjątkiem Ctrl+J i C) jest już w programie Excel wykorzystywanych do innych zadań Jednym z rozwiązańjest przypisywanie do makra połączenia klawiszy Ctrl+Shift+A aż do Ctrl+Shift+Z. Aby przypisać do makra skrót Ctrl+Shift+A, w polu skrótu wpisujemy **A**. Zwróćmy uwagę, że w tym przypadku rozróżniana jest wielkość liter – fakt, że trzeba użyć klawisza Shift, aby uzyskać wielką literę, sprawia, że staje się on częścią skrótu.

**Ostrzeżenie** Przypisanie skrótów do makr zmienia dotychczasowe przypisania klawiszy. Jeśli na przykład przypiszemy do makra kombinację Ctrl+C, po naciśnięciu tych klawiszy Excel uruchomi makro, zamiast wykonać standardowe kopiowanie.

**Uwaga** Punkt „Inne sposoby uruchamiania makr" w rozdziale 25 przedstawia inne sposoby uruchamiania makr, takie jak wykorzystanie hiperłączy, przycisków, kształtów, dodatkowych ikon we wstążce i wiele innych.

W oknie dialogowym **Rejestrowanie makra** można zdecydować o tym, gdzie makro ma być zapisane po zarejestrowaniu. Dostępne opcje to: **Skoroszyt makr osobistych**, **Nowy skoroszyt** oraz **Ten skoroszyt**. Zaleca się, by zapisywać makra związane z danym skoroszytem przy użyciu opcji **Ten skoroszyt**.

Skoroszyt makr osobistych (Personal.xlsm) nie jest widoczny – jest tworzony, jeśli do zapisu wybierzemy opcję **Skoroszyt makr osobistych**. Funkcja ta służy do zapisywania makr w skoroszycie otwieranym automatycznie podczas uruchamiania programu Excel, dzięki czemu od razu będzie można korzystać z makra. Po uruchomieniu programu Excel skoroszyt jest ukrywany. Aby wyświetlić ten skoroszyt, należy wybrać opcję **Odkryj** okno na karcie **Widok**.

**Wkazówka** Nie zaleca się używania skoroszytu makr osobistych dla wszystkich zapisywanych makr. Należy zapisywać w nim tylko te makra, które pomagają w wykonywaniu zadań ogólnego przeznaczenia, a nie zadań, które wykonywane są dla określonego arkusza lub skoroszytu.

W czwartym polu okna dialogowego Rejestrowanie makra wpisujemy opis. Opis ten zostanie dodany jako komentarz na początku makra.

Po wybraniu lokalizacji, w której ma być zapisane makro, należy kliknąć **OK**. Teraz trzeba zarejestrować makro. Na przykład w aktywnej komórce wpisujemy **Hello World** i naciskamy Ctrl+Enter, by zaakceptować wpis i pozostać w tej samej komórce. Po zakończeniu rejestrowania makra klikamy ikonę **Zatrzymaj rejestrowanie** na karcie Deweloper.

**Wkazówka** Ikona Zatrzymaj rejestrowanie jest również dostępna w lewym dolnym rogu okna programu Excel i wygląda jak niewielki niebieski kwadrat. Znajduje się z prawej strony słowa *Gotowy* na pasku stanu. Użycie tego przycisku może być czasami wygodniejsze niż powrót karty Deweloper. Po zarejestrowaniu pierwszego makra w tym obszarze zazwyczaj widoczna jest ikona Rejestruj makro.

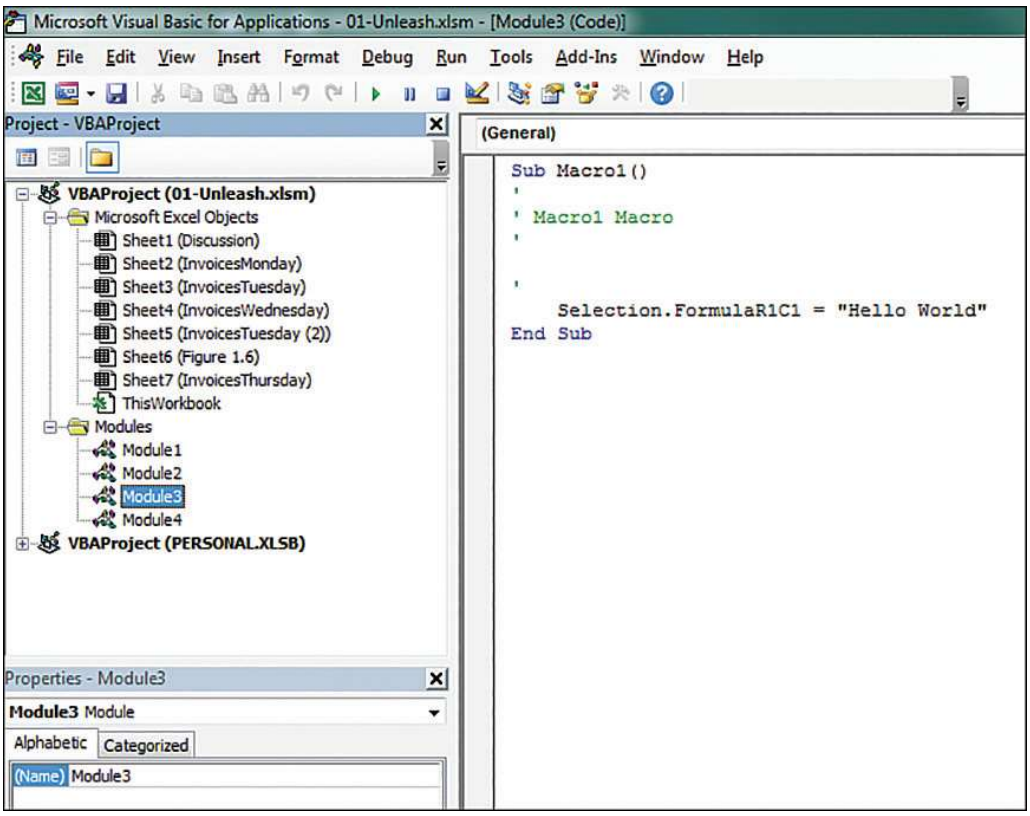
### Uruchamianie makra

Jeśli przypisaliśmy klawisz skrótu do makra, możemy je uruchomić poprzez wciśnięcie tej kombinacji klawiszy. Makra można także przypisywać do przycisków na wstążce lub na pasku narzędzi Szybki dostęp, do kontrolek formularza, do obiektów graficznych lub też możemy je uruchamiać z paska narzędzi Visual Basic (patrz punkt „Inne sposoby uruchamiania makr" w rozdziale 25).

## Poznajemy edytor Visual Basic

Narzędzie VB Editor umożliwia edycję zarejestrowanego makra. Aby uruchomić edytor, naciskamy klawisze Alt+F11 lub używamy ikony Visual Basic na karcie Deweloper*.

Na rysunku 1.5 pokazano przykład typowego ekranu edytora VB, na którym widoczne są trzy okna: Project Explorer, Properties i Programming (okno eksploratora projektu, właściwości i kodu). Nie należy się przejmować tym, że na komputerze mamy trochę inny ekran, niż okno prezentowane na rysunku, ponieważ podczas omawiania edytora dowiemy się, jak wyświetlać potrzebne okna.




**Rysunek 1.5** Okno VB Editor

* Narzędzie VB Editor dostępne jest tylko w angielskiej wersji interfejsu użytkownika.

## Ustawienia narzędzia VB Editor

Szereg opcji edytora VB umożliwia dostosowanie narzędzia, by łatwiej było nam pisać makra.

Klika przydatnych ustawieńudostępnionych jest na karcie **Tools > Options > Editor**. Wszystkie opcje, poza jedną, mają odpowiednie domyślne ustawienie. Jedno ustawienie wymaga podjęcia decyzji ze strony użytkownika. Chodzi o opcję **Require Variable Declaration** (wymagaj deklarowania zmiennych). Zgodnie z ustawieniami domyślnymi Excel nie wymaga deklarowania zmiennych. Mój współautor preferuje to ustawienie, ponieważ pozwala ono oszczędzić czas podczas tworzenia programów. Ja jednak jestem odmiennego zdania i wolę zmienić to ustawienie, aby zadeklarowanie zmiennej było wymagane. Powoduje to zatrzymanie kompilatora, gdy napotka zmienną, której nie rozpoznaje, co zmniejsza liczbę pomyłek w nazwach. Włączenie lub wyłączenie tej opcji zależy od osobistych preferencji użytkownika.

## Project Explorer

Okno eksploratora projektu wymienia wszystkie otwarte skoroszyty i załadowane dodatki. Jeśli klikniemy ikonę + obok pozycji VBAProject, zobaczymy folder Microsoft Excel Objects. Mogą tam być również foldery formularzy, modułów klas i modułów standardowych. Każdy folder zawiera jeden lub kilka komponentów.

Kliknięcie komponentu prawym przyciskiem myszy i wybranie polecenia **View Code** lub dwukrotne kliknięcie komponentu powoduje wyświetlenie kodu w oknie Programming. Wyjątek stanowią formularze użytkowników – w tym przypadku dwukrotne kliknięcie powoduje wyświetlenie formularza w widoku projektu (Design).

Aby wyświetlić okno Project Explorer, należy z menu wybrać polecenie **View > Project Explorer**, nacisnąć klawisze Ctrl+R lub znaleźć i kliknąć dziwną ikonę eksploratora projektu na pasku narzędzi, poniżej menu Tools, wciśniętą pomiędzy ikony Design Mode i Properties Window.

Aby wstawić moduł, klikamy projekt prawym przyciskiem myszy, wybieramy polecenie **Insert**, a następnie wskazujemy typ modułu. Poniżej wymieniono dostępne moduły:

- **Obiekty Microsoft Excel** Domyślnie projekt zawiera moduły arkuszy dla każdego arkusza w skoroszycie oraz jeden moduł ThisWorkbook (Ten skoroszyt). Kod specyficzny dla arkusza, taki jak kontrolki czy zdarzenia związane z arkuszem, umieszczany jest w odpowiadającym mu arkuszu. Kod obsługi zdarzeń skoroszytu umieszczony jest w module ThisWorkbook. Więcej informacji na temat zdarzeń można znaleźć w rozdziale 7 „Programowanie zdarzeń”.
- **Formularze** Program Excel pozwala zaprojektować własne formularze do komunikowania się z użytkownikami. Więcej informacji na temat formularzy można znaleźć w rozdziale 10 „Formularze użytkownika – wprowadzenie”.

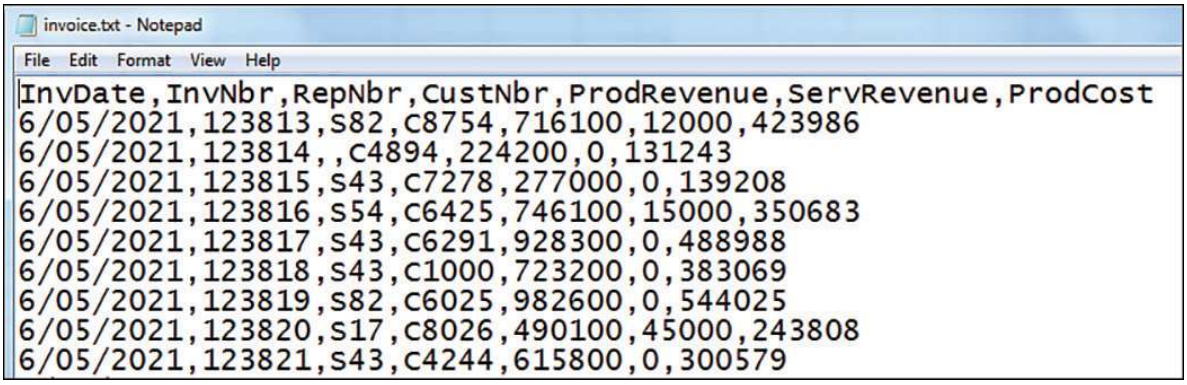
- **Moduły** Podczas rejestrowania makra program Excel automatycznie tworzy moduł, w którym umieszcza kod. Większość tworzonego przez nas kodu umieszczana jest w modułach tego typu.
- **Moduły klas** Moduły klas to sposób na tworzenie własnych obiektów. Moduły klas pozwalają również na współdzielenie fragmentów kodu pomiędzy programistami bez konieczności poznania sposobu działania kodu. Więcej informacji na temat modułów klas znaleźć można w rozdziale 9 „Tworzenie niestandardowych obiektów i i kolekcji".

### Okno Properties

Okno Properties umożliwia edycję właściwości różnych komponentów, takich jak arkusze, skoroszyty, moduły i kontrolki formularzy. Lista właściwości może być różna, w zależności od wybranego komponentu. Aby wyświetlić to okno, z menu wybieramy polecenie **View > Properties Window**, wciskamy klawisz F4 lub klikamy ikonę okna Properties na pasku narzędzi.

## Mankamenty rejestratora makr

Załóżmy, że pracujemy w dziale księgowości. Każdego dnia otrzymujemy plik tekstowy z systemu firmy z listą wszystkich faktur wystawionych poprzedniego dnia. W tym pliku tekstowym poszczególne pola są rozdzielone przecinkami. Kolumny w pliku to InvDate (Data faktury), InvNbr (Numer faktury), RepNbr (Numer sprzedawcy), CustNbr (Numer klienta), ProdRevenue (Dochód produktu), ServRevenue (Dochód usługi) oraz ProdCost (Koszt produktu) (patrz rysunek 1.6).




**Rysunek 1.6** Plik Invoice.txt

Każdego ranka ręcznie importujemy ten plik do programu Excel. Dodajemy wiersz podsumowania, pogrubiamy nagłówki, a następnie drukujemy raport i przekazujemy go kilku menedżerom.

Wydaje się, że jest to prosty proces, idealnie nadający się do wykorzystania rejestratora makr. Jednak z powodu pewnych problemów z rejestratorem makr nasze pierwsze próby zautomatyzowania tego procesu nie muszą zakończyć się sukcesem. Poniższa analiza przypadku pokazuje, jak pokonać te problemy.

## Studium przypadku: Przygotowanie do rejestracji makra

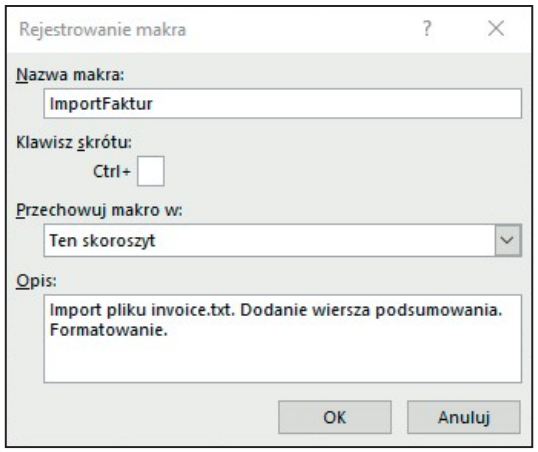
Zadanie opisane w poprzednim fragmencie idealnie nadaje się do utworzenia makra. Przed zarejestrowaniem makra należy jednak przeanalizować wykonywane czynności. W naszym przypadku są to następujące operacje:

1. Kliknięcie menu **Plik** i wybranie polecenia **Otwórz**.
2. Przejście do folderu, w którym jest zapisany plik invoice.txt.
3. Z listy rozwijanej **Pliki typu** wybranie opcji **Wszystkie pliki**.
4. Wybranie pliku invoice.txt.
5. Kliknięcie przycisku **Otwórz**.
6. W oknie **Kreator importu tekstu – krok 1 z 3**, w sekcji **Typ danych źródłowych** zaznaczenie opcji **Rozdzielany**.
7. Kliknięcie przycisku **Dalej**.
8. W oknie **Kreator importu tekstu – krok 2 z 3**, w ramce **Ograniczniki** usunięcie zaznaczenia pola wyboru **Tabulator** i zaznaczenie pola wyboru **Przecinek**.
9. Kliknięcie **Dalej**.
10. W oknie **Kreator importu tekstu – krok 3 z 3**, w ramce **Format danych w kolumnie** wybranie opcji **Data: MDR**.
11. Klikniecie przycisku **Zakończ**, aby zaimportować plik.
12. Naciśnięcie klawiszy Ctrl i strzałka w dół, by przejść do ostatniego wiersza danych.
13. Ponownie wciśnięcie strzałki w dół, by przejść do wiersza podsumowania.
14. Wpisanie słowa **Razem**.
15. Cztery razy naciśnięcie klawisza strzałka w prawo, by przejść do kolumny E wiersza podsumowania.
16. Kliknięcie przycisku **Autosumowanie**, a następnie naciśnięcie klawiszy Ctrl+Enter, by dodać podsumowanie do kolumny ProdRevenue i jednocześnie pozostać w tej komórce.
17. Przeciągnięcie uchwytu automatycznego wypełniania z kolumny E do kolumny G w celu skopiowania formuły podsumowującej do kolumn F i G.
18. Zaznaczenie pierwszego wiersza i kliknięcie ikony **Pogrubienie** na karcie **Narzędzia główne**, by wyróżnić czcionkę nagłówków.

19. Zaznaczenie wiersza podsumowania i kliknięcie ikony **Pogrubienie**, by wyróżnić zawartość tego wiersza.
20. Wciśnięcie Ctrl+A, by zaznaczyć bieżący obszar.
21. Na karcie **Narzędzia główne** wybranie narzędzia **Formatuj** i opcji **Autodopasowanie szerokości kolumn**.

Po zebraniu czynności, które należy wykonać, możemy przystąpić do rejestracji pierwszego makra. Otwieramy pusty skoroszyt i zapisujemy go na przykład pod nazwą MacroToImportInvoices.xlsm. Na karcie Deweloper klikamy teraz przycisk **Zarejestruj makro**.

Domyślna nazwa makra w oknie **Rejestrowanie makra** to ***Makro1***. Zmieniamy tę nazwę na bardziej opisową, przykładowo **ImportInvoices**. Upewniamy się, że do zapisu makra wybrana jest opcja **Ten skoroszyt**. Ponieważ później przyda się łatwy sposób uruchamiania tego makra, w polu **Klawisz skrótu** wpisujemy literę **i**. W polu **Opis** dodajemy krótki opis działania makra (rysunek 1.7). Po wykonaniu tych czynności klikamy **OK**.




**Rysunek 1.7** Przed zarejestrowaniem makra należy wypełnić pola w oknie dialogowym Rejestrowanie makra.

## Rejestrowanie makra

Teraz rejestrator makr zarejestruje każdy nasz ruch. Z tego względu należy starać się wykonywać wszystkie czynności po kolei, bez żadnych dodatkowych działań. Jeśli przypadkowo przejdziemy do kolumny F, a następnie z powrotem do kolumny E w celu wprowadzenia pierwszej sumy, zarejestrowane makro będzie codziennie „ślepo" powielało tę samą pomyłkę. Zarejestrowane makra działają szybko, ale nie warto przyglądać się, jak za każdym razem wszystkie nasze pomyłki są powtarzane. Należy zatem uważnie wykonać wszystkie działania niezbędne do utworzenia raportu. Po wykonaniu ostatniej czynności klikamy przycisk **Zatrzymaj rejestrowanie** na karcie Deweloper.

## Analiza kodu w oknie programowania

Spróbujmy przyjrzeć się kodowi, który przed chwilą zarejestrowaliśmy w analizie przypadku. Nic nie szkodzi, jeśli na razie kod ten nie jest zrozumiały.

Aby otworzyć edytor VB, naciskamy klawisze Alt+F11. Dla w projekcie VBA (MacroToImportInvoices.xlsm) odnajdujemy komponent Module1. Trzeba go kliknąć prawym przyciskiem myszy i wybrać polecenie **View Code**. Warto zwrócić uwagę, że niektóre wiersze rozpoczynają się od apostrofu; są to komentarze, które są ignorowane przez program. Rejestrator makr rozpoczyna makra od kilku komentarzy. Do tego celu wykorzystuje opis, który wprowadziliśmy w oknie dialogowym Rejestrowanie makra. Komentarz dotyczący klawisza skrótu ma przypominać, jaki klawisz skrótu wybraliśmy dla makra.

---

**Uwaga** Komentarz *nie* powoduje przypisania klawisza skrótu. Jeśli zmodyfikujemy komentarz i wprowadzimy tam ciąg Ctrl+J, nie spowoduje to zmiany klawisza skrótu przypisanego do makra. W celu zmiany klawisza skrótu należy zmienić ustawienie w oknie dialogowym Makro w Excelu lub uruchomić poniższy kod:

```
Application.MacroOptions Macro:="ImportFaktur", _
    Description:="", ShortcutKey:="j"
```

---

Zarejestrowane makro jest zazwyczaj dość schludne (rysunek 1.8). W każdym wierszu kodu (poza wierszami komentarza) jest wcięcie o szerokości czterech znaków. W celu kontynuacji wiersza kodu w następnym wierszu należy na końcu wiersza umieścić spację i znak podkreślenia. Nie zapominajmy o spacji przez znakiem podkreślania – jej brak spowoduje błąd.

---

**Uwaga** Ze względu na fizyczne rozmiary tej książki w pojedynczym wierszu nie zmieści się 100 znaków. Dlatego wiersze będą dzielone przy szerokości około 78 znaków, by mogły zmieścić się na stronie. Z tych też powodów makro zarejestrowane na komputerze Czytelnika może się nieco różnić od prezentowanego w książce.

---

Zwróćmy uwagę, że poniższe sześć wierszy zarejestrowanego kodu to w rzeczywistości tylko jeden wiersz, podzielony dla poprawy czytelności:

```
Workbooks.OpenText Filename:= "jakaś_ścieżka\invoice.txt", Origin:=437, _
    StartRow:=1, DataType:=xlDelimited, TextQualifier:=xlDoubleQuote, _
    ConsecutiveDelimiter:=False, Tab:=False, Semicolon:=False, Comma:=True, _
    Space:=False, Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
    Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), _
    TrailingMinusNumbers:=True
```

Uwzględniając, że powyższy kod to jeden wiersz, widzimy, że rejestrator makr zarejestrował proces składający się z 21 kroków w 14 wierszach kodu. Robi wrażenie.

```
Sub ImportInvoice()
'
' ImportInvoice Macro
' Import Invoice.txt. Add Total Row. Format.
'
' Keyboard Shortcut: Ctrl+i
'
    Workbooks.OpenText Filename:="G:\2016VBA\SampleFiles\invoice.txt", Origin:= _
        437, StartRow:=1, DataType:=xlDelimited, TextQualifier:=xlDoubleQuote, _
        ConsecutiveDelimiter:=False, Tab:=False, Semicolon:=False, Comma:=True _
        , Space:=False, Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
        Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), TrailingMinusNumbers _
        :=True
    Selection.End(xlDown).Select
    Range("A11").Select
    ActiveCell.FormulaR1C1 = "Total"
    Range("E11").Select
    Selection.FormulaR1C1 = "=SUM(R[-9]C:R[-1]C)"
    Selection.AutoFill Destination:=Range("E11:G11"), Type:=xlFillDefault
    Range("E11:G11").Select
    Rows("1:1").Select
    Selection.Font.Bold = True
    Rows("11:11").Select
    Selection.Font.Bold = True
    Selection.CurrentRegion.Select
    Selection.Columns.AutoFit
End Sub
```

**Rysunek 1.8** Zarejestrowane makro ma schludny wygląd i estetyczne wcięcia

**Uwaga** Każde działanie wykonywane za pomocą interfejsu użytkownika programu Excel może odpowiadać jednemu lub kilku wierszom zarejestrowanego kodu. Niektóre działania mogą generować nawet kilkanaście wierszy kodu.

## Testowanie każdego makra

Zawsze warto sprawdzać makra. W celu przetestowania nowego makra powrócimy do standardowego interfejsu programu Excel, naciskając klawisze Alt+F11. Zamykamy plik invoice.txt bez zapisywania zmian. Arkusz MacroToImportInvoices.xlsm jest w dalszym ciągu otwarty.

Naciskamy klawisze Ctrl+I, by uruchomić zarejestrowane makro. Makro powinno zadziałać dobrze, jeśli prawidłowo wykonaliśmy poszczególne kroki procedury. Dane zostały zaimportowane, podsumowania dodane, formatowanie zastosowane, a szerokość kolumn powiększona. Wydaje się, że to doskonałe rozwiązanie (rysunek 1.9).

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | **InvDate** | **InvNbr** | **RepNbr** | **CustNbr** | **ProdRevenue** | **ServRevenue** | **ProdCost** |
| 2 | 6/5/2025 | 123813 | S82 | C8754 | 716100 | 12000 | 423986 |
| 3 | 6/5/2025 | 123814 | | C4894 | 224200 | 0 | 131243 |
| 4 | 6/5/2025 | 123815 | S43 | C7278 | 277000 | 0 | 139208 |
| 5 | 6/5/2025 | 123816 | S54 | C6425 | 746100 | 15000 | 350683 |
| 6 | 6/5/2025 | 123817 | S43 | C6291 | 928300 | 0 | 488988 |
| 7 | 6/5/2025 | 123818 | S43 | C1000 | 723200 | 0 | 383069 |
| 8 | 6/5/2025 | 123819 | S82 | C6025 | 982600 | 0 | 544025 |
| 9 | 6/5/2025 | 123820 | S17 | C8026 | 490100 | 45000 | 243808 |
| 10 | 6/5/2025 | 123821 | S43 | C4244 | 615800 | 0 | 300579 |
| 11 | **Total** | | | | **5703400** | **72000** | **3005589** |
| 12 | | | | | | | |

**Rysunek 1.9** Makro formatuje dane w arkuszu

## Uruchomienie tego samego makra innego dnia generuje nieoczekiwane wyniki

Po przetestowaniu makra należy je zapisać, by można go było użyć następnego dnia. Następnego dnia po przyjściu do pracy otrzymaliśmy nowy plik invoice.txt z systemu.

---

**Wskazówka** Dla tych Czytelników, którzy w trakcie lektury korzystają z plików przykładowych dla tej książki, oto kroki, które należy wykonać, aby spróbować zaimportować dane dla innego dnia:

1. Zamknij plik Invoice.txt w programie Excel.
2. W Eksploratorze Windows zmień nazwę Invoice.txt na Invoice1.txt.
3. Przemianuj plik Invoice2.txt na Invoice.txt.
4. Wróć do Excela i arkusza MacroToImportInvoices.xlsm.
5. Naciśnij Ctrl+I, aby uruchomić makro dla większego zbioru danych.

---

Otwieramy makro, wciskamy Ctrl+I, by je uruchomić i… mamy katastrofę. W pliku z 5 czerwca jest 9 faktur, a 6 czerwca faktur jest 17. Zarejestrowane makro „ślepo" dodało wiersz z podsumowaniem w wierszu 11, ponieważ właśnie w tym wierszu umieściliśmy podsumowanie w momencie, gdy było ono rejestrowane (rysunek 1.10).

Problem wynika stąd, że rejestrator makr zapisał wszystkie działania w trybie odwołań bezwzględnych (domyślnie). Jako alternatywę dla stosowania domyślnego stanu rejestratora w kolejnym rozdziale omówię rejestrowanie względnych odwołań, co powinno nas zbliżyć do pożądanego rozwiązania.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **InvDate** | **InvNbr** | **RepNbr** | **CustNbr** | **ProdRevenue** | **ServRevenue** | **ProdCost** | |
| 2 | 6/6/2025 | 123829 | S21 | C8754 | 21000 | 0 | 9875 | |
| 3 | 6/6/2025 | 123830 | S45 | C3390 | 188100 | 0 | 85083 | |
| 4 | 6/6/2025 | 123831 | S54 | C2523 | 510600 | 0 | 281158 | |
| 5 | 6/6/2025 | 123832 | S21 | C5519 | 86200 | 0 | 49967 | |
| 6 | 6/6/2025 | 123833 | S45 | C3245 | 800100 | 0 | 388277 | |
| 7 | 6/6/2025 | 123834 | S54 | C7796 | 339000 | 0 | 195298 | |
| 8 | 6/6/2025 | 123835 | S21 | C1654 | 161000 | 0 | 90761 | |
| 9 | 6/6/2025 | 123836 | S45 | C6460 | 275500 | 10000 | 146341 | |
| 10 | 6/6/2025 | 123837 | S54 | C5143 | 925400 | 0 | 473515 | |
| 11 | **Total** | **123838** | **S21** | **C7868** | **3306900** | **10000** | **1720275** | |
| 12 | 6/6/2025 | 123839 | S45 | C3310 | 890200 | 0 | 468333 | |
| 13 | 6/6/2025 | 123840 | S54 | C2959 | 986000 | 0 | 528980 | |
| 14 | 6/6/2025 | 123841 | S21 | C8361 | 94400 | 0 | 53180 | |
| 15 | 6/6/2025 | 123842 | S45 | C1842 | 36500 | 55000 | 20696 | |
| 16 | 6/6/2025 | 123843 | S54 | C4107 | 599700 | 0 | 276718 | |
| 17 | 6/6/2025 | 123844 | S21 | C5205 | 244900 | 0 | 143393 | |
| 18 | 6/6/2025 | 123845 | S45 | C7745 | 63000 | 0 | 35102 | |
| 19 | 6/6/2025 | 123846 | S54 | C1730 | 212600 | 0 | 117787 | |
| 20 | 6/6/2025 | 123847 | S21 | C6292 | 974700 | 0 | 478731 | |
| 21 | 6/6/2025 | 123848 | S45 | C2008 | 327700 | 0 | 170968 | |
| 22 | 6/6/2025 | 123849 | S54 | C4096 | 30700 | 0 | 18056 | |

**Rysunek 1.10** Makro miało dodawać podsumowanie na końcu pliku, ale rejestrator tak zarejestrował makro, że podsumowanie zawsze wyświetla się w wierszu 11.

## Możliwe rozwiązanie: wykorzystywanie odwołań względnych podczas rejestrowania

Domyślnie rejestrator makr rejestruje wszystkie działania jako *odwołania bezwzględne*. Jeśli podczas rejestrowania przechodzimy do wiersza 11, w momencie uruchamiania makro zawsze będzie przechodziło do wiersza 11. Bardzo rzadko takie działanie będzie odpowiednie w przypadku zmiennej liczby wierszy danych. Lepszym rozwiązaniem podczas rejestrowania jest wykorzystanie odwołań względnych.

W makrach zarejestrowanych w trybie odwołań bezwzględnych są zapisywane rzeczywiste adresy wskaźnika komórki (przykładowo A11). W makrach zarejestrowanych w trybie odwołań względnych zapisane są informacje o tym, że wskaźnik komórki powinien przemieścić się o określoną liczbę wierszy i kolumn w odniesieniu do bieżącego położenia. Na przykład, jeśli wskaźnik komórki znajduje się w komórce A1, kod `ActiveCell.Offset(16,1).Select` przeniesie wskaźnik do komórki B17, czyli do komórki, która znajduje się o 16 wierszy w dół i jedną kolumnę w prawo.

Pomimo że rejestrowanie w trybie odwołań względnych jest odpowiednie dla większości sytuacji, czasami podczas rejestrowania może zachodzić jednak potrzeba użycia odwołania bezwzględnego. Ten przykład jest dobrą ilustracją takiej sytuacji: Po dodaniu wiersza podsumowania do zestawu danych trzeba powrócić do wiersza 1. Jeśli

w trybie odwołań względnych po prostu klikniemy wiersz 1, program Excel zarejestruje operację zaznaczenia wiersza, który znajduje się 10 wierszy powyżej wiersza bieżącego. Taka operacja zadziała w przypadku pierwszego pliku invoice.txt, ale nie będzie działać w przypadku dłuższego lub krótszego pliku. Poniżej wymieniono dwie metody obejścia problemu:

- Wyłączenie rejestrowania w trybie odwołań względnych, kliknięcie wiersza 1, a następnie ponowne włączenie rejestrowania względnego.
- Zachowanie włączonego rejestrowania w trybie odwołań względnych. Następnie wyświetlamy okno dialogowe **Przechodzenie do** poprzez naciśnięcie klawisza F5. Wpisujemy A1 i klikamy OK. Okno dialogowe **Przechodzenie do** rejestrowane jest w typowy sposób i następuje przejście do adresu bezwzględnego, nawet jeśli włączone jest rejestrowanie w trybie odwołań względnych. Odmiana tej metody wykorzystana została w kolejnej analizie przypadku.

Spróbujmy ponownie przeprowadzić analizę tego samego przypadku – tym razem z wykorzystaniem odwołań względnych. Rozwiązanie będzie znacznie bliższe rozwiązaniu poprawnemu.

**Uwaga** jeśli stosujemy pliki przykładów, należy wykonać następujące działania:

1. Zamknąć plik invoice.txt w programie Excel.
2. Zmienić nazwę pliku invoice.txt na invoice2.txt.
3. Zmianić nazwę pliku invoice1.txt na invoice.txt.
4. Wrócić do skoroszytu MacroToImportInvoices.xlsm.

## Studium przypadku: Rejestrowanie makra przy użyciu odwołań względnych

Spróbujmy zarejestrować makro jeszcze raz, tym razem z wykorzystaniem odwołań względnych.

Na karcie Deweloper wybieramy przycisk Użyj odwołań względnych, by zmienić tryb rejestrowania. Ustawienie to jest aktywne, dopóki nie zostanie wyłączone lub do zamknięcia programu Excel.

W skoroszycie MacroToImportInvoices.xlsm rejestrujemy nowe makro poprzez wybranie polecenia **Zarejestruj makro** na karcie Deweloper. Nadajemy mu opisową nazwę, na przykład **ImportInvoicesRelative** i przypisujemy inny klawisz skrótu, np. Ctrl+Shift+J.

Należy powtórzyć kroki 1 do 11 z poprzedniego przykładu, by zaimportować plik, a następnie wykonać poniższe instrukcje:

1. Naciśnij Ctrl+strzałka w dół, by przejść do ostatniego wiersza danych.
2. Ponownie naciśnij klawisz strzałki w dół, by przejść do wiersza podsumowania.
3. Wpisz słowo **Razem**.
4. Cztery razy naciśnij strzałkę w prawo, by przejść do kolumny E wiersza podsumowania.
5. Przytrzymaj klawisz Shift, naciskając jednocześnie dwa razy strzałkę w prawo, by zaznaczyć zakres E11:G11.
6. Kliknij przycisk Autosumowanie.
7. Naciśnij klawisze Shift+spacja, by zaznaczyć cały wiersz. Naciśnij klawiszy Ctrl+B, by do wiersza zastosować formatowanie: pogrubienie.
8. Naciśnij F5, by wyświetlić okno dialogowe **Przechodzenie do**.
9. W oknie dialogowym Przechodzenie do, wpisz **A1:G1** i kliknij przycisk **OK**. Nawet jeśli włączone jest rejestrowanie względne, każdy ruch wykonywany za pośrednictwem okna dialogowego Przechodzenie do zostanie zarejestrowana jako odwołanie bezwzględne.
10. Kliknij ikonę Pogrubienie, by sformatować nagłówki.
11. Naciśnij Ctrl+*, by zaznaczyć wszystkie dane w bieżącym regionie.
12. Na karcie Narzędzia główne wybierz opcję **Autodopasowanie szerokości kolumn** (w menu **Formatuj**).
13. Zatrzymaj rejestrowanie.

Naciskamy klawisze Alt+F11, by przejść do edytora VB i przejrzeć kod. Nowe makro widoczne jest w Module1, poniżej poprzedniego makra.

Jeśli pomiędzy rejestrowaniem pierwszego i drugiego makra zamkniemy i ponownie uruchomimy program Excel, nowo zarejestrowany modułu otrzyma nazwę Module2.

```
Sub ImportInvoicesRelative()
' ImportInvoicesRelative Macro
' Import. Total Row. Format.
' Keyboard Shortcut: Ctrl+J
Workbooks.OpenText Filename:="C:\data\invoice.txt", _
  Origin:= 437, StartRow:=1, DataType:=xlDelimited, _
  TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _
  Tab:=False, Semicolon:=False, Comma:=True, Space:=False, _
  Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
  Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), _
  Array(7, 1)), TrailingMinusNumbers:=True
Selection.End(xlDown).Select
ActiveCell.Offset(1, 0).Range("A1").Select
ActiveCell.FormulaR1C1 = "Total"
ActiveCell.Offset(0, 4).Range("A1:C1").Select
Selection.FormulaR1C1 = "=SUM(R[-9]C:R[-1]C)"
ActiveCell.Rows("1:1").EntireRow.Select
```

```
ActiveCell.Activate
Selection.Font.Bold = True
Application.Goto Reference:="R1C1:R1C7"
Selection.Font.Bold = True
Selection.CurrentRegion.Select
Selection.Columns.AutoFitSelection.Font.Bold = True
End Sub
```

W celu przetestowania makra zamykamy plik invoice.txt bez zapisywania zmian, a następnie uruchamiamy makro za pomocą klawiszy Ctrl+Shift+J. Wszystko wygląda poprawnie – uzyskaliśmy te same wyniki za pomocą makra utworzonego przez rejestrator.

Następny test polega na sprawdzeniu, czy program zadziała następnego dnia, w którym może być więcej wierszy. Jeśli wykorzystywaliśmy pliki przykładów, zamykamy plik invoice.txt w programie Excel. Zmieniamy nazwę pliku invoice.txt na invoice2.txt i zmieniamy nazwę pliku invoice4.txt na invoice.txt.

Otwieramy plik MacroToImportInvoices i uruchamiamy makro za pomocą skrótu Ctrl+Shift+J. Tym razem wszystko wygląda dobrze, a sumy umieszczone są na właściwych miejscach. Spójrzmy na rysunek 1.11. Czy jest tam coś niezwykłego?

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **InvDate** | **InvNbr** | **RepNbr** | **CustNbr** | **ProdRevenue** | **ServRevenue** | **ProdCost** | |
| 2 | 6/6/2025 | 123829 | S21 | C8754 | 21000 | 0 | 9875 | |
| 3 | 6/6/2025 | 123830 | S45 | C3390 | 188100 | 0 | 85083 | |
| 4 | 6/6/2025 | 123831 | S54 | C2523 | 510600 | 0 | 281158 | |
| 5 | 6/6/2025 | 123832 | S21 | C5519 | 86200 | 0 | 49967 | |
| 6 | 6/6/2025 | 123833 | S45 | C3245 | 800100 | 0 | 388277 | |
| 7 | 6/6/2025 | 123834 | S54 | C7796 | 339000 | 0 | 195298 | |
| 8 | 6/6/2025 | 123835 | S21 | C1654 | 161000 | 0 | 90761 | |
| 9 | 6/6/2025 | 123836 | S45 | C6460 | 275500 | 10000 | 146341 | |
| 10 | 6/6/2025 | 123837 | S54 | C5143 | 925400 | 0 | 473515 | |
| 11 | 6/6/2025 | 123838 | S21 | C7868 | 148200 | 0 | 75700 | |
| 12 | 6/6/2025 | 123839 | S45 | C3310 | 890200 | 0 | 468333 | |
| 13 | 6/6/2025 | 123840 | S54 | C2959 | 986000 | 0 | 528980 | |
| 14 | 6/6/2025 | 123841 | S21 | C8361 | 94400 | 0 | 53180 | |
| 15 | 6/6/2025 | 123842 | S45 | C1842 | 36500 | 55000 | 20696 | |
| 16 | 6/6/2025 | 123843 | S54 | C4107 | 599700 | 0 | 276718 | |
| 17 | 6/6/2025 | 123844 | S21 | C5205 | 244900 | 0 | 143393 | |
| 18 | 6/6/2025 | 123845 | S45 | C7745 | 63000 | 0 | 35102 | |
| 19 | 6/6/2025 | 123846 | S54 | C1730 | 212600 | 0 | 117787 | |
| 20 | 6/6/2025 | 123847 | S21 | C6292 | 974700 | 0 | 478731 | |
| 21 | 6/6/2025 | 123848 | S45 | C2008 | 327700 | 0 | 170968 | |
| 22 | 6/6/2025 | 123849 | S54 | C4096 | 30700 | 0 | 18056 | |
| 23 | **Total** | | | | **2584200** | **55000** | **1314631** | |

**Rysunek 1.11** Rezultat uzyskany po uruchomieniu makra w trybie odwołań względnych.

O ile nie będziemy uważni, moglibyśmy wydrukować ten raport i zanieść menedżerowi. A następnie okazałoby się, że możemy mieć kłopoty. Gdy spojrzymy na komórkę E23, widzimy, że Excel wstawił tam zielony trójkąt, by zwrócić uwagę na tę komórkę.

Kiedy przeniesiemy wskaźnik położenia do komórki E23, obok komórki pojawi się wskaźnik ostrzeżenia, który informuje, że w formuła nie dołącza przylegających komórek. Przyglądając się paskowi formuły, zauważymy, że makro posumowało tylko wiersze od 14 do 22. Ani rejestrowanie z użyciem odwołań względnych, ani z użyciem odwołań bezwzględnych nie jest na tyle „sprytne", by powielić logikę przycisku Autosumowanie.

Załóżmy, że w danym dniu mamy mniej faktur. Program Excel „nagrodziłby" nas nielogiczną formułą =SUM(E6:E1048574), jak ilustruje to rysunek 1.12. Ponieważ formuła ta byłaby w komórce E7, w pasku stanu pojawiłoby się ostrzeżenie o odwołaniu cyklicznym.

**Uwaga** Aby wypróbować to samemu, trzeba zamknąć plik invoice.txt w programie Excel. Następnie zmieniamy nazwę pliku invoice.txt na invoice2.txt i zmieniamy nazwę pliku invoice4.txt na invoice.txt.

E7 =SUM(E6:E1048574)

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **InvDate** | **InvNbr** | **RepNbr** | **CustNbr** | **ProdRevenue** | **ServRevenue** | **ProdCost** | |
| 2 | 6/7/2025 | 123850 | | C1654 | 161000 | 0 | 90761 | |
| 3 | 6/7/2025 | 123851 | | C6460 | 275500 | 10000 | 146341 | |
| 4 | 6/7/2025 | 123852 | | C5143 | 925400 | 0 | 473515 | |
| 5 | 6/7/2025 | 123853 | | C7868 | 148200 | 0 | 75700 | |
| 6 | 6/7/2025 | 123854 | | C3310 | 890200 | 0 | 468333 | |
| 7 | **Total** | | | | **0** | **0** | **0** | |

**Rysunek 1.12** Wynik działania makro w trybie odwołań względnych przy mniejszej liczbie rekordów faktur.

Każdy, kto próbował używać rejestratora makr, prawdopodobnie napotkał problemy podobne do tych, które opisano w analizie ostatnich dwóch przypadków. Chociaż jest to frustrujące, pozytywne jest to, że rejestrator makr pozwala zrealizować 95% zadania związanego z utworzeniem poprawnego makra.

Naszym zadaniem jest rozpoznanie miejsc, w których działanie rejestratora makr może zawieść, a następnie zmodyfikowanie kodu VBA w celu poprawienia jednego czy kilku wierszy i uzyskania poprawnie działającego makra. Korzystając z własnej inwencji, możemy tworzyć ciekawe makra, które przyspieszą nasze codzienne działania.

## Podczas rejestrowania nigdy nie używaj przycisku Autosumowanie lub Szybka analiza

Opisany problem dotyczący rejestrowania funkcji Autosumowanie można jednak rozwiązać. Trzeba jedynie zapamiętać, że rejestrator makr nigdy poprawnie nie zarejestruje działania przycisku Autosumowanie.

Jeśli rozpoczniemy działanie w komórce E99 i klikniemy przycisk Autosumowanie, program Excel zacznie skanowanie od komórki E98 w górę do czasu, aż odnajdzie komórkę tekstową, pustą komórkę lub formułę. Następnie zaproponuje formułę, która sumuje wszystkie komórki pomiędzy komórką bieżącą a tą znalezioną.

Rejestrator makr rejestruje jednak określony wynik tego wyszukiwania w momencie, w którym zarejestrowano makro. Zamiast zarejestrować coś w stylu „wykonaj zwykłą logikę przycisku Autosumowanie", rejestrator makr wstawia pojedynczy wiersz kodu sumujący poprzednie 98 komórek.

W podobny sposób działa funkcja Szybka analiza:

1. Zaznaczamy zakres E2:G99.
2. Otwieramy ikonę **Szybka analiza**, która wyświetlana jest poniżej prostokątnego zaznaczenia z prawej jego strony.
3. Wybieramy opcję **Suma** i uzyskujemy poprawne sumy w wierszu 100.

Rejestrator makr na sztywno zakoduje formuły, by zawsze wyświetlane były w wierszu 100 i zawsze sumowały wiersze od 2 do 99.

Dość dziwacznym obejściem tego problemu jest wpisanie funkcji SUMA, która wykorzystuje połączenie odwołań względnych i bezwzględnych. Jeśli wpiszemy `=SUMA(E$2:E10)` w czasie działania rejestratora makr, program Excel poprawnie doda kod, który zawsze będzie sumował komórki, począwszy od ustalonego wiersza 2 w dół do względnego odwołania, które jest bezpośrednio nad komórką bieżącą.

Poniżej zaprezentowano uzyskany kod:

```
Sub FormatInvoice3()
' Makro FormatInvoice3
' Importowanie. Podsumowanie. Formatowanie.
' Skrót klawiszowy: Ctrl+K
Workbooks.OpenText Filename:="C:\Data\invoice.txt", _
  Origin:=437, StartRow:=1, DataType:=xlDelimited, _
  TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _
  Tab:=False, Semicolon:=False, Comma:=True , Space:=False, _
  Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
  Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), _
  Array(7, 1)), TrailingMinusNumbers:=True
Selection.End(xlDown).Select
ActiveCell.Offset(1, 0).Range("A1").Select
ActiveCell.FormulaR1C1 = "Total"
ActiveCell.Offset(0, 4).Range("A1").Select
Selection.FormulaR1C1 = "=SUM(R2C:R[-1]C)"
```

```
Selection.AutoFill Destination:=ActiveCell.Range("A1:C1"), _
  Type:=xlFillDefault
ActiveCell.Range("A1:C1").Select
ActiveCell.Rows("1:1").EntireRow.Select
ActiveCell.Activate
Selection.Font.Bold = True
Application.Goto Reference:="R1C1:R1C7"
Selection.Font.Bold = True
Selection.CurrentRegion.Select
Selection.Columns.AutoFit
End Sub
```

To trzecie makro działa poprawnie dla zbioru danych dowolnego rozmiaru.

## Cztery wskazówki dotyczące rejestratora makr

Bardzo rzadko zdarza się sytuacja, by zarejestrowane makro zadziałało w 100%. Będziemy jednak znacznie bliżej tego celu, jeśli zastosujemy się do czterech wskazówek prezentowanych poniżej.

### Wskazówka 1: Włącz ustawienie Użyj odwołań względnych

Microsoft powinien nadać temu ustawieniu status ustawienia domyślnego. Włącz ustawienie i nie wyłączaj go podczas rejestrowania makr.

### Wskazówka 2: Używaj specjalnych klawiszy nawigacyjnych do przechodzenia na koniec zbioru danych

Jeśli jesteś na początku zbioru danych i chcesz przejść do ostatniej komórki z danymi, naciśnij Ctrl+strzałka w dół lub naciśnij klawisz End, a następnie strzałkę w dół.

Aby przejść do ostatniej kolumny w bieżącym wierszu zbioru danych, naciśnij klawisze Ctrl+strzałka w prawo lub naciśnij klawisz End, a następnie strzałkę w prawo. Dzięki użyciu tych klawiszy nawigacyjnych można przechodzić na koniec zbioru danych niezależnie od tego, z ilu wierszy lub kolumn składa się zbiór danych określonego dnia.

Używaj klawiszy Ctrl+*, by zaznaczać bieżący region wokół komórki aktywnej. Zakładając, że dane nie zawierają pustych wierszy lub kolumn, ta kombinacja klawiszy powoduje zaznaczenie całego zestawu danych.

**Uwaga** Trzeba pamiętać, że znak * (gwiazdka) z klawiatury „podstawowej" jest uzyskiwany z wciśniętym klawiszem Shift – skrót ten można zatem zapisać jako Ctrl+Shift+8. Używamy znaku *, gdyż ten sam efekt można uzyskać, naciskając znak * na klawiaturze numerycznej (bez Shift).

### Wskazówka 3: Podczas rejestrowania makra nigdy nie używaj przycisku Autosumowanie

Rejestrator makr nie potrafi zarejestrować „istoty" działania przycisku Autosumowanie. Zamiast tego koduje „na sztywno" formułę, wynikającą z naciśnięcia tego przycisku. Formuła ta nie zadziała poprawnie, jeśli zbiór danych będzie zawierał mniej lub więcej rekordów.

Zamiast tego należy wpisać formułę z pojedynczym znakiem dolara, na przykład `=SUMA(E$2:E10)`. W takim przypadku rejestrator makr zarejestruje pierwsze odwołanie E$2 jako ustalone i rozpocznie określać zakres sumowania bezpośrednio pod nagłówkami w pierwszym wierszu. Zakładając, że bieżącą komórką jest E11, rejestrator makr rozpozna E10 jako odwołanie względne wskazujące bezpośrednio powyżej bieżącej komórki.

### Wskazówka 4: Próbuj rejestrować makra innymi metodami, jeśli zastosowana metoda nie działa

Excel zwykle oferuje wiele sposobów wykonywania tych samych zadań. Jeśli napotkamy błędny kod generowany przez jeden sposób, warto spróbować innego. Ponieważ w zespole menedżerów projektów programu Excel jest 16 osób, prawdopodobnie każda metoda była programowana przez inną grupę. W jednej z analiz przypadków w tym rozdziale, jedno zadanie wiązało się z zastosowaniem do wszystkich komórek funkcji Autodopasowanie szerokości kolumn. Niektóre osoby mogą nacisnąć klawisze Ctrl+A, by zaznaczyć wszystkie komórki, a inne mogą nacisnąć klawisze Ctrl+*. Począwszy od wersji Excel 2007 przy włączonym trybie odwołań względnych kod generowany przez naciśnięcie klawiszy Ctrl+A nie działa. Kod Ctrl+* jest bardzo stary i nadal działa we wszystkich przypadkach.

## Następne kroki

W tym rozdziale dowiedzieliśmy się, jak rejestrować makra i je uruchamiać w celu automatyzowania podstawowych zadań w Excelu. Trzeba jednak mieć na uwadze, że choć rejestrator makr jest pomocnym punktem startowym, często tworzy kod, który jest niewydajny lub po prostu błędny.

W rozdziale 2 „Skoro to BASIC, dlaczego nie wygląda znajomo?", przeanalizujemy trzy makra, które zarejestrowaliśmy w tym rozdziale i postaramy się je zrozumieć. Kiedy już nauczymy się czytać kod VBA, naturalnym krokiem będzie poprawienie zarejestrowanego kodu lub napisanie poprawnego od początku. Warto przeczytać kolejny rozdział – wkrótce przekonamy się, że VBA to dobre rozwiązanie i nauczymy się pisać dobrze działający kod.