Katalog procedur i funkcji w aplikacji AgrX

Aplikacja kontroli poprawności gromadzenia danych w plikach gromadzenia danych

Krzysztof Sługocki, k.slugocki@protonmail.com

14.09.2023

Komentarz do kodu procedury RaportInfoAWN_1

```
Sub RaportInfoAWN_1()
' Deklaracja zmiennych
Dim str_Info As String
Dim i_n As Integer
Dim i As Integer
Dim j As Integer
Dim d_msz As Double
Dim d_moswz As Double
Dim i_Lawn As Integer
Dim str_Grupa As String
Pliki.Select
' Wygeneruj listę plików
ListaPlikow_2
' Wstaw bieżącą datę i godzinę
Range("C5").FormulaR1C1 = "=TERAZ()"
str_Info = Left(Range("C5").Value, 16)
' Utwórz nagłówek raportu
CreateReport ("Raport z: " & str_Info)
' Wywołaj podprogram awnX
'awnX
PerformDataAnalysis
' Dodaj linię separatora
CreateReport ("-------------------------------------------------------
-------")
' Wybierz arkusz "RAP"
RAP.Select
' Wyczyść dane zaczynając od komórki B6
ClearWorksheetRange RAP, "B6"
' Odczytaj dane tekstowe
ReadTextFile
' Wygeneruj ostateczny raport
'OstatniRaport
UpdateLastReport
' Wyświetl wiadomość dla użytkownika
MsgBox "Raport został dodatkowo zapisany w pliku [raport.txt]." _
& vbCrLf & "Zawarte jest tam zestawienie wszystkich dotychczasowych
raportów." _
& vbCrLf & "Plik raportu znajduje się w folderze Aplikacji." _
& vbCrLf & vbCrLf & "http://art30a.ucoz.pl/,
k.slugocki@protonmail.com" _
```

```
, vbInformation + vbSystemModal, "Raport analizy poprawności
gromadzenia danych"
End Sub
```

Wyjaśnienia

W tym kodzie mamy podprogram o nazwie "RaportInfoAWN_1", który wykonuje różne zadania związane z generowaniem raportów i analizą danych. Poniżej znajduje się wyjaśnienie poszczególnych linii kodu:

- `Dim str_Info As String`: Deklaruje zmienną `str_Info` jako typu String, która będzie przechowywać informacje o bieżącej dacie i godzinie.
- `Dim i_n As Integer, i As Integer, j As Integer`: Deklaruje zmienne `i_n`, i i `j` jako typ Integer, które będą używane jako zmienne pomocnicze w pętlach i iteracjach.
- `Dim d_msz As Double, d_moswz As Double`: Deklaruje zmienne `d_msz` i `d_moswz` jako typ Double, które będą przechowywać wartości liczbowe.
- `Dim i_Lawn As Integer, str_Grupa As String`: Deklaruje zmienne `i_Lawn` jako typ Integer i `str_Grupa` jako typ String, które będą przechowywać informacje o liczbie plików i grupie.
- `Pliki.Select`: Wybiera arkusz o nazwie "Pliki".
- `ListaPlikow_2`: Wywołuje podprogram "ListaPlikow_2", który generuje listę plików.
- `Range("C5").FormulaR1C1 = "=TERAZ()"`: Wstawia formułę "=TERAZ()" do komórki C5, co spowoduje wyświetlenie bieżącej daty i godziny.
- `str_Info = Left(Range("C5").Value, 16)`: Przypisuje zmiennej `str_Info` wartość lewej części (pierwsze 16 znaków) zawartości komórki C5, co umożliwia uzyskanie bieżącej daty i godziny w formacie tekstowym.
- `CreateReport ("Raport z: " & str_Info)`: Wywołuje podprogram "CreateReport" i przekazuje jako argument tekst "Raport z: " połączony z wartością zmiennej `str_Info`. Ten podprogram tworzy nagłówek raportu.
- `PerformDataAnalysis`: Wywołuje podprogram "PerformDataAnalysis", który wykonuje analizę danych.
- `CreateReport ("-------------------------------------------------------")`: Wywołuje podprogram "CreateReport" i przekazuje jako argument linię separatora, która zostanie dodana do raportu.
- `RAP.Select`: Wybiera arkusz o nazwie "RAP".
- `ClearWorksheetRange RAP, "B6"`: Wywołuje podprogram "ClearWorksheetRange" i przekazuje jako argumenty arkusz "RAP" oraz komórkę B6. Ten podprogram usuwa dane zaczynając od komórki B6.
- `ReadTextFile`: Wywołuje podprogram "ReadTextFile", który odczytuje dane tekstowe.
- `UpdateLastReport`: Wywołuje podprogram "UpdateLastReport", który generuje ostateczny raport.
- `MsgBox "Raport został dodatkowo zapisany w pliku [raport.txt]." _ . . .`: Wyświetla wiadomość dla użytkownika, informującą o zapisaniu raportu w pliku "raport.txt" oraz zawierającą inne informacje dotyczące raportu.

Ten kod jest częścią większego programu napisanego w języku VBA. Przez dodawanie komentarzy do kodu, programista może łatwiej zrozumieć, co robią poszczególne linie kodu i jakie są ich efekty. Komentarze są szczególnie przydatne dla innych programistów, którzy mogą pracować nad tym samym kodem w przyszłości.

Komentarz do kodu procedury GenerateReport

```
Sub GenerateReport(strInfo As String)
' Generuje raport i dołącza informacje do pliku raportu.
Dim reportFolder As String
Dim reportPath As String
Dim fileNumber As Integer
' Pobierz folder raportu z określonej komórki.
reportFolder = Pliki.Range("C4").Value
' Skonstruuj pełną ścieżkę pliku raportu.
reportPath = reportFolder & "\raport.txt"
' Pobierz wolny numer pliku.
fileNumber = FreeFile
On Error GoTo ErrorHandler
' Otwórz plik raportu w trybie dołączania.
Open reportPath For Append As #fileNumber
' Dołącz podane informacje do pliku raportu.
Print #fileNumber, strInfo
' Zamknij plik raportu.
Close #fileNumber
Exit Sub
ErrorHandler:
' Obsłuż błędy, np. plik nie znaleziony lub problem
z uprawnieniami.
MsgBox "Błąd: " & Err.Number & vbCrLf & Err.Description,
vbExclamation, "Błąd"
End Sub
```

Wyjaśnienia

Ten kod VBA generuje raport i dołącza informacje do pliku raportu. Poniżej znajduje się wyjaśnienie dodanych komentarzy:

1. `' Generuje raport i dołącza informacje do pliku raportu.` – Ten komentarz opisuje ogólną funkcję procedury `GenerateReport`. Informuje nas, że ta procedura generuje raport i dołącza informacje do pliku raportu.
2. `' Pobierz folder raportu z określonej komórki.` – Ten komentarz wyjaśnia, że zmienna `reportFolder` jest pobierana z określonej komórki w arkuszu o nazwie "Pliki". Ta komórka zawiera ścieżkę do folderu, w którym ma być zapisany raport.
3. `' Skonstruuj pełną ścieżkę pliku raportu.` – Ten komentarz informuje nas, że zmienna `reportPath` jest skonstruowana przez połączenie ścieżki folderu raportu z nazwą pliku "raport.txt". Ta pełna ścieżka wskazuje miejsce, w którym zostanie zapisany raport.
4. `' Pobierz wolny numer pliku.` – Ten komentarz wyjaśnia, że zmienna `fileNumber` jest używana do przechowywania numeru wolnego pliku. Jest to potrzebne do otwarcia pliku raportu w trybie dołączania.
5. `' Otwórz plik raportu w trybie dołączania.` – Ten komentarz informuje nas, że plik raportu jest otwierany w trybie dołączania. Oznacza to, że nowe informacje będą dołączane na końcu istniejącego pliku, zamiast go nadpisywać.

6. `' Dołącz podane informacje do pliku raportu.` – Ten komentarz opisuje, że podane informacje (przechowywane w zmiennej `strInfo`) są dołączane do pliku raportu za pomocą instrukcji `Print #fileNumber, strInfo`. To jest miejsce, w którym właściwe informacje są dodawane do raportu.
7. `' Zamknij plik raportu.` – Ten komentarz mówi nam, że plik raportu jest zamykany za pomocą instrukcji `Close #fileNumber`. Po zamknięciu pliku raportu, wszystkie zmiany zostaną zapisane.
8. `' Obsłuż błędy, np. plik nie znaleziony lub problem z uprawnieniami.` – Ten komentarz informuje nas, że w przypadku wystąpienia błędu (np. plik nie został znaleziony lub wystąpił problem z uprawnieniami), zostanie wyświetlone okno dialogowe z informacją o błędzie. Komunikat zawiera numer błędu i opis błędu.

Komentarz do kodu funkcji FindLastA

Funkcja FindLastA jest używana do znalezienia ostatniego używanego wiersza w określonej kolumnie. W celu lepszej czytelności i zrozumienia kodu, dodano komentarz, który opisuje działanie funkcji.

```
Function FindLastA(col As Long) As Long
' Znajduje ostatni używany wiersz w określonej kolumnie.
' col: Numer kolumny, w której ma być wyszukiwany ostatni
używany wiersz.
Dim lastRow As Long
On Error Resume Next
lastRow = Cells.Find("*", Cells(1, col), xlFormulas, xlWhole,
xlByRows, xlPrevious).Row
On Error GoTo 0
FindLastA = lastRow
End Function
```

Wyjaśnienia

Kod funkcji FindLastA jest dość prosty, ale dodanie komentarza pomaga w zrozumieniu jego działania. Komentarz znajduje się na początku funkcji i opisuje, że funkcja ta służy do znalezienia ostatniego używanego wiersza w określonej kolumnie.

Dodatkowo, komentarz zawiera informację o parametrze "col", który jest numerem kolumny, w której ma być wyszukiwany ostatni używany wiersz. Dzięki temu komentarzowi programista może łatwo zrozumieć, jak używać tej funkcji i jakie argumenty przekazać.

Kod funkcji korzysta z metody Find, która wyszukuje ostatni używany wiersz w określonej kolumnie. Wartość "*" oznacza, że funkcja ma wyszukać dowolną wartość w kolumnie. Parametr Cells(1, col) określa, że wyszukiwanie ma rozpocząć się od pierwszego wiersza i określonej kolumny. Następnie, parametry xlFormulas, xlWhole, xlByRows i xlPrevious określają, jak ma być wykonane wyszukiwanie.

W przypadku wystąpienia błędu, instrukcja On Error Resume Next pozwala na kontynuację wykonywania kodu bez przerywania. Natomiast instrukcja On Error GoTo 0 przywraca normalne zachowanie obsługi błędów.

Na końcu funkcji, zmienna lastRow przechowuje numer ostatniego używanego wiersza, który jest zwracany jako wynik funkcji.

Dzięki dodanemu komentarzowi, programiści, którzy będą korzystać z tej funkcji, będą mieli łatwiejsze zadanie w zrozumieniu jej działania i poprawnym jej użyciu.

Komentarz do kodu procedury UpdateLastReport

```
Sub UpdateLastReport() ' Aktualizuje ostatni raport na arkuszu.
Dim lastRow As Long
Dim i As Long
' Znajdź ostatni wiersz w kolumnie A z danymi.
lastRow = FindLastA(1)
' Przejdź przez wiersze, zaczynając od wiersza 6 do
ostatniego wiersza.
For i = 6 To lastRow
' Sprawdź, czy zawartość w kolumnie B pasuje do nagłówka raportu.
If Left(Cells(i, 2).Value, 25) = "Raport z: " &
Left(Range("C2").Value, 15) Then
' Skopiuj zawartość z kolumny B z dopasowanego wiersza do
ostatniego wiersza.
Range("B" & i - 1 & ":B" & lastRow).Select
Selection.Copy
' Wklej skopiowaną zawartość do komórki C7.
Range("C7").Select
ActiveSheet.Paste
' Wyczyść zawartość w kolumnie B od wiersza 7 do
ostatniego wiersza.
Range("B7:B" & lastRow).Select
Selection.ClearContents
' Powróć do komórki A1, aby zakończyć operację.
Range("A1").Select
Exit For ' Wyjdź z pętli po znalezieniu pierwszego dopasowania.
End If
Next i
End Sub
```

Wyjaśnienia

W powyższym kodzie mamy procedurę o nazwie "UpdateLastReport", która służy do aktualizacji ostatniego raportu na arkuszu. Kod ten wykonuje następujące czynności:

1. Znajduje ostatni wiersz w kolumnie A z danymi, korzystając z funkcji "FindLastA".
2. Przechodzi przez wiersze, zaczynając od wiersza 6 do ostatniego wiersza.
3. Sprawdza, czy zawartość w kolumnie B pasuje do nagłówka raportu, porównując pierwsze 25 znaków z wartością komórki C2.
4. Jeśli pasuje, kopiuję zawartość z kolumny B z dopasowanego wiersza do ostatniego wiersza.
5. Wkleja skopiowaną zawartość do komórki C7.
6. Wyczyść zawartość w kolumnie B od wiersza 7 do ostatniego wiersza.
7. Powraca do komórki A1, aby zakończyć operację.

Ten kod jest przykładem prostego skryptu VBA, który automatyzuje proces aktualizacji ostatniego raportu na arkuszu. Komentarze w kodzie są dodane, aby pomóc zrozumieć poszczególne kroki i działanie kodu.

Komentarz do kodu procedury PerformSheetAnalysis

```
Sub PerformSheetAnalysis(sheetName As String)
' Wykonuje analizę danych na określonym arkuszu.
Dim infoMessage As String
' Aktywuje określony arkusz.
Sheets(sheetName).Activate
' Wykonuje analizę danych dla określonego arkusza.
AnalizaXMinEt onlyFileName
' Wyświetla komunikat informacyjny o zakończeniu analizy.
'infoMessage = "Analiza poprawności gromadzenia danych w pliku " &
onlyFileName & " - " & sheetName & " została zakończona." _
& vbCrLf & vbCrLf &
"http://art30a.ucoz.pl, k.slugocki@protonmail.com"
'MsgBox infoMessage, vbInformation + vbSystemModal, "Zakończono
analizę danych"
End Sub
```

Wyjaśnienia

W powyższym kodzie mamy procedurę `PerformSheetAnalysis`, która wykonuje analizę danych na określonym arkuszu.

W pierwszym komentarzu `Wykonuje analizę danych na określonym arkuszu.` wyjaśniamy, że ta procedura jest odpowiedzialna za analizę danych na określonym arkuszu.

W drugim komentarzu `Aktywuje określony arkusz.` wyjaśniamy, że ta linia kodu aktywuje określony arkusz, aby można było na nim wykonać analizę danych.

W trzecim komentarzu `Wykonuje analizę danych dla określonego arkusza.` wyjaśniamy, że ta linia kodu wywołuje funkcję `AnalizaXMinEt` dla określonego arkusza, aby przeprowadzić analizę danych.

W czwartym komentarzu `Wyświetla komunikat informacyjny o zakończeniu analizy.` wyjaśniamy, że ta linia kodu jest zakomentowana, ale pierwotnie miała na celu wyświetlenie komunikatu informującego o zakończeniu analizy danych.

Komentarze są użyteczne, ponieważ pomagają innym programistom zrozumieć, co robi dany kod i jak go używać.

Komentarz do kodu procedury OpenWorkbookDialog

```
Sub OpenWorkbookDialog() ' Otwiera okno dialogowe w celu wyboru
pliku skoroszytu.
Dim selectedFileName As Variant
Dim folderPath As String
'Dim onlyFileName As String
Dim fileFilter As String
Dim fileIndex As Integer
Dim dialogTitle As String
' Przełącz się na arkusz "Pliki".
Sheets("Pliki").Select
' Pobierz ścieżkę folderu z komórki C4 w arkuszu "Pliki".
folderPath = Range("C4").Value
' Zmień bieżący dysk i katalog na określony folder.
ChDrive folderPath
ChDir folderPath
' Zdefiniuj filtr plików, indeks pliku i tytuł dialogu.
fileFilter = "Pliki do gromadzenia danych (awn*.xlsx), awn*.xlsx,
Pliki Excel (*.xl*), *xl.*"
fileIndex = 1
dialogTitle = "Wybierz plik do analizy poprawności
gromadzenia danych"
' Wyświetl okno dialogowe otwierania pliku.
selectedFileName = Application.GetOpenFilename(fileFilter,
fileIndex, dialogTitle)
' Wyodrębnij nazwę pliku bez ścieżki.
onlyFileName = FullNameToFileName(selectedFileName)
' Sprawdź, czy został wybrany plik.
If selectedFileName <> "False" And selectedFileName <> "" Then
' Poinformuj użytkownika i otwórz wybrany skoroszyt.
MsgBox "Analiza danych jest przeprowadzana <<w tle>>: " &
onlyFileName _
& vbCrLf & "Po zakończeniu analizy poprawności gromadzenia danych,
plik danych należy zamknąć, o ile nie zostanie zamknięty
programowo." _
& vbCrLf & vbCrLf & "http://art30a.ucoz.pl,
k.slugocki@protonmail.com" _
, vbInformation + vbSystemModal, "Analiza danych"
Workbooks.Open fileName:=selectedFileName
Else
' Powiadom użytkownika, że nie został wybrany żaden plik.
MsgBox "Żaden plik [awn*.xlsx] nie został wybrany.", vbExclamation,
"Brak wybranego pliku"
End If
' Aktywuj skoroszyt "AgrX.xlsm".
Windows("AgrX.xlsm").Activate
End Sub
```

Wyjaśnienia

W powyższym kodzie VBA mamy procedurę o nazwie "OpenWorkbookDialog", która otwiera okno dialogowe, umożliwiające wybór pliku skoroszytu. Kod ten jest komentowany w celu lepszego zrozumienia jego działania.

Pierwszy komentarz w kodzie znajduje się przy deklaracji zmiennej "selectedFileName". Komentarz informuje, że zmienna ta przechowuje nazwę wybranego pliku skoroszytu.

Kolejny komentarz znajduje się przy deklaracji zmiennej "folderPath". Komentarz informuje, że zmienna ta przechowuje ścieżkę folderu, którą pobieramy z komórki C4 w arkuszu "Pliki".

Następnie mamy komentarz przy deklaracji zmiennej "fileFilter". Komentarz informuje, że zmienna ta definiuje filtr plików, który będzie wykorzystywany w oknie dialogowym otwierania pliku.

Kolejny komentarz znajduje się przy deklaracji zmiennej "fileIndex". Komentarz informuje, że zmienna ta określa indeks pliku, który zostanie wybrany w oknie dialogowym.

Ostatni komentarz w tej sekcji znajduje się przy deklaracji zmiennej "dialogTitle". Komentarz informuje, że zmienna ta przechowuje tytuł okna dialogowego otwierania pliku.

Następnie mamy komentarz przed wywołaniem metody "Application.GetOpenFilename". Komentarz informuje, że ta metoda wyświetla okno dialogowe otwierania pliku i zwraca nazwę wybranego pliku.

Kolejny komentarz znajduje się przed wywołaniem funkcji "FullNameToFileName". Komentarz informuje, że ta funkcja służy do wyodrębnienia nazwy pliku bez ścieżki.

Następnie mamy warunek "If selectedFileName <> "False" And selectedFileName <> "" Then", który sprawdza, czy został wybrany plik. Komentarz informuje, że jeśli warunek jest spełniony, zostanie wyświetlone powiadomienie dla użytkownika i otwarty zostanie wybrany skoroszyt.

Ostatni komentarz w kodzie znajduje się przed wywołaniem metody "Windows("AgrX.xlsm").Activate". Komentarz informuje, że ta metoda aktywuje skoroszyt o nazwie "AgrX.xlsm".

Dzięki komentarzom w kodzie, łatwiej jest zrozumieć, co robi poszczególna część kodu i jakie są jej zadania. Komentarze są ważne, ponieważ pomagają innym programistom w zrozumieniu kodu i ułatwiają jego utrzymanie w przyszłości.

Komentarz do kodu procedury PerformDataAnalysis

```
Sub PerformDataAnalysis()
' Performs data analysis on a selected workbook.
Dim folderPath As String
Dim infoMessage As String
On Error GoTo ErrorHandler
' Open a workbook using a dialog box.
OpenWorkbookDialog
' Get the selected folder path.
folderPath = Pliki.Range("C4").Value
' Display an information message to the user.
infoMessage = "Po zaakceptowaniu tego komunikatu rozpocznie się
analiza poprawności gromadzenia danych danych." _
& vbCrLf & "Analiza może potrwać kilkanaście sekund." _
& vbCrLf & "Zakończenie analizy zostanie zasygnalizowane
komunikatem." _
& vbCrLf & vbCrLf &
"http://art30a.ucoz.pl, k.slugocki@protonmail.com"
MsgBox infoMessage, vbInformation + vbSystemModal, "Data Analysis"
' Disable screen updating to speed up the analysis.
Application.ScreenUpdating = False
' Open the selected workbook and perform analysis.
Workbooks.Open fileName:=folderPath & "\" & onlyFileName
GenerateReport onlyFileName
PerformSheetAnalysis "nd"
PerformSheetAnalysis "nm"
PerformSheetAnalysis "nk"
' Close the workbook without saving changes.
Application.DisplayAlerts = False
ActiveWindow.Close SaveChanges:=False
Application.DisplayAlerts = True
' Re-enable screen updating.
Application.ScreenUpdating = True
' Display a completion message to the user.
MsgBox "Analiza danych w pliku " & onlyFileName & " została
zakończona." _
& vbCrLf & vbCrLf & "http://art30a.ucoz.pl,
k.slugocki@protonmail.com" _
, vbInformation + vbSystemModal, "Data Analysis Completed"
Exit Sub
ErrorHandler:
' Handle errors by displaying a user-friendly message.
MsgBox "Error #" & Err.Number & vbCrLf & "Sprawdź nazwę pliku lub
wystąpił inny problem:" _
& vbCrLf & Err.Description, vbCritical + vbSystemModal, "Error"
End Sub
```

Wyjaśnienia

W powyższym kodzie VBA została dodana komentarz do każdej sekcji kodu w celu lepszego zrozumienia jego działania. Komentarze są używane do opisania poszczególnych kroków i funkcji, które są wykonywane w kodzie.

Komentarze są ważne, ponieważ pomagają innym programistom zrozumieć kod i jego zamiar. Dzięki nim łatwiej jest zidentyfikować, co dany fragment kodu robi i jak działa. Komentarze są również przydatne podczas późniejszych modyfikacji kodu, ponieważ ułatwiają zrozumienie jego struktury i logiki.

W powyższym kodzie zostały dodane komentarze w języku polskim, które opisują poszczególne sekcje kodu. Komentarze zawierają informacje o tym, co dany fragment kodu robi, jakie są jego cele i jakie są oczekiwane rezultaty.

Dodanie komentarzy do kodu jest dobrym nawykiem programistycznym, który pomaga w utrzymaniu czytelności i zrozumiałości kodu. Dzięki komentarzom inni programiści mogą łatwiej zrozumieć kod i współpracować nad nim.

Komentarz do kodu procedury ReadTextFile

```
Sub ReadTextFile()
' Odczytuje zawartość pliku tekstowego i umieszcza ją
w arkuszu kalkulacyjnym.
Dim folderPath As String
Dim reportPath As String
Dim lineCounter As Integer
Dim currentLine As String
' Pobierz ścieżkę folderu z komórki C4 w arkuszu Pliki.
folderPath = Pliki.Range("C4").Value
' Skonstruuj pełną ścieżkę do pliku tekstowego raportu.
reportPath = folderPath & "\raport.txt"
' Zainicjuj licznik linii.
lineCounter = 0
' Sprawdź, czy plik raportu istnieje.
If Dir(reportPath) <> "" Then
' Otwórz plik raportu do odczytu.
Open reportPath For Input As #1
' Odczytaj każdą linię z pliku i umieść ją w arkuszu RAP.
Do Until EOF(1)
lineCounter = lineCounter + 1
Line Input #1, currentLine
RAP.Cells(5 + lineCounter, 2).Value = currentLine
Loop
' Zamknij plik raportu.
Close #1
Else
' Jeśli plik raportu nie istnieje, wyświetl komunikat.
MsgBox "Plik raportu nie istnieje w określonym folderze.",
vbExclamation, "Plik nie znaleziony"
End If
End Sub
```

Wyjaśnienia

Ten kod VBA odczytuje zawartość pliku tekstowego i umieszcza ją w arkuszu kalkulacyjnym. Oto wyjaśnienie dodanych komentarzy:

1. `' Odczytuje zawartość pliku tekstowego i umieszcza ją w arkuszu kalkulacyjnym.` – Ten komentarz opisuje ogólną funkcję kodu, czyli odczytanie pliku tekstowego i umieszczenie jego zawartości w arkuszu kalkulacyjnym.
2. `' Pobierz ścieżkę folderu z komórki C4 w arkuszu Pliki.` – Ten komentarz wyjaśnia, że kod pobiera ścieżkę folderu z komórki C4 w arkuszu Pliki. Ta ścieżka będzie używana do skonstruowania pełnej ścieżki do pliku raportu.
3. `' Skonstruuj pełną ścieżkę do pliku tekstowego raportu.` – Ten komentarz informuje, że kod konstruuje pełną ścieżkę do pliku raportu, łącząc ścieżkę folderu z nazwą pliku raportu.
4. `' Zainicjuj licznik linii.` – Ten komentarz mówi nam, że kod inicjalizuje licznik linii, który będzie używany do śledzenia, na której linii pliku znajduje się aktualnie odczytywana linia.

5. `' Sprawdź, czy plik raportu istnieje.` – Ten komentarz opisuje, że kod sprawdza, czy plik raportu istnieje w określonym folderze. Jeśli plik istnieje, kod będzie kontynuował odczyt.
6. `' Otwórz plik raportu do odczytu.` – Ten komentarz informuje, że kod otwiera plik raportu do odczytu.
7. `' Odczytaj każdą linię z pliku i umieść ją w arkuszu RAP.` – Ten komentarz opisuje, że kod odczytuje każdą linię z pliku raportu i umieszcza ją w arkuszu RAP. Każda linia jest umieszczana w kolejnych wierszach arkusza, zaczynając od wiersza 5.
8. `' Zamknij plik raportu.` – Ten komentarz mówi nam, że kod zamyka plik raportu po zakończeniu odczytu.
9. `' Jeśli plik raportu nie istnieje, wyświetl komunikat.` – Ten komentarz informuje, że jeśli plik raportu nie istnieje, zostanie wyświetlony komunikat o braku pliku.

Dzięki tym komentarzom kod jest czytelniejszy i łatwiejszy do zrozumienia dla innych programistów, którzy mogą go przeglądać i modyfikować w przyszłości.

Komentarz do kodu procedury ClearWorksheetRange

```
Sub ClearWorksheetRange(ByRef targetWorksheet As Worksheet, ByVal
anchorCellAddress As String) ' Czyści zawartość określonego zakresu
w arkuszu.
Dim lastRow As Long
Dim lastCol As Long
' Aktywuje docelowy arkusz.
targetWorksheet.Activate
' Znajduje ostatni używany wiersz w każdej kolumnie do kolumny 51.
lastRow = 0
For lastCol = 1 To 51
Dim colLastRow As Long
colLastRow = Cells(Rows.Count, lastCol).End(xlUp).Row
If colLastRow > lastRow Then
lastRow = colLastRow
End If
Next lastCol
' Wybiera zakres do wyczyszczenia.
targetWorksheet.Range(anchorCellAddress).Select
targetWorksheet.Range(Selection, Selection.Rows(lastRow)).Select
targetWorksheet.Range(Selection, Selection.Columns(256)).Select
' Czyści zawartość wybranego zakresu.
Selection.ClearContents
' Powrót do komórki kotwicy.
targetWorksheet.Range(anchorCellAddress).Select
End Sub
```

Wyjaśnienia

W powyższym kodzie mamy procedurę o nazwie "ClearWorksheetRange", która służy do czyszczenia zawartości określonego zakresu w arkuszu. Kod ten jest napisany w języku VBA (Visual Basic for Applications) i może być używany w programie Excel.

Komentarze w kodzie są używane do opisywania poszczególnych kroków i funkcji w celu ułatwienia zrozumienia kodu przez inne osoby. W tym przypadku, komentarze zostały dodane, aby wyjaśnić działanie poszczególnych linii kodu.

Pierwszy komentarz został dodany przed procedurą, aby opisać jej funkcję – czyścić zawartość określonego zakresu w arkuszu.

Kolejne komentarze zostały dodane przed poszczególnymi liniami kodu, aby wyjaśnić, co ta linia robi. Na przykład, komentarz przed liniami kodu "targetWorksheet.Activate" mówi nam, że ta linia aktywuje docelowy arkusz.

Komentarze są ważne, ponieważ pomagają innym programistom zrozumieć kod i szybko zlokalizować konkretne funkcje. Dzięki nim kod staje się bardziej czytelny i łatwiejszy do utrzymania.

Komentarz do kodu procedury CreateReport

```
Sub CreateReport(strInfo As String) ' Tworzy plik raportu i dołącza
podane informacje.
Dim folderPath As String
Dim reportPath As String
Dim fileNumber As Integer
' Pobierz ścieżkę folderu z określonej komórki.
folderPath = Pliki.Range("C4").Value
' Skonstruuj ścieżkę pliku raportu.
reportPath = folderPath & "raport.txt"
' Sprawdź, czy ścieżka folderu jest prawidłowa.
If Len(Dir(folderPath, vbDirectory)) = 0 Then
MsgBox "Nieprawidłowa ścieżka dostępu.", vbExclamation, "Błąd"
Exit Sub
End If
' Sprawdź, czy plik raportu już istnieje.
If Len(Dir(reportPath)) = 0 Then
' Jeśli nie istnieje, utwórz plik raportu.
fileNumber = FreeFile
Open reportPath For Output As #fileNumber
Close #fileNumber
End If
' Dołącz podane informacje do pliku raportu.
fileNumber = FreeFile
Open reportPath For Append As #fileNumber
Print #fileNumber, strInfo
Close #fileNumber
End Sub
```

Wyjaśnienia

W powyższym kodzie mamy procedurę CreateReport, która tworzy plik raportu i dołącza podane informacje. Poniżej znajduje się wyjaśnienie dodanych komentarzy:

1. Komentarz: Tworzy plik raportu i dołącza podane informacje. Wyjaśnienie: Ten komentarz opisuje ogólną funkcję procedury CreateReport, czyli tworzenie pliku raportu i dołączanie informacji.
2. Komentarz: Pobierz ścieżkę folderu z określonej komórki. Wyjaśnienie: Ten komentarz wskazuje, że procedura pobiera ścieżkę folderu z określonej komórki w arkuszu Excel.
3. Komentarz: Skonstruuj ścieżkę pliku raportu. Wyjaśnienie: Ten komentarz informuje, że procedura tworzy ścieżkę pliku raportu, łącząc ścieżkę folderu z nazwą pliku "raport.txt".
4. Komentarz: Sprawdź, czy ścieżka folderu jest prawidłowa. Wyjaśnienie: Ten komentarz wskazuje, że procedura sprawdza, czy podana ścieżka folderu istnieje. Jeśli nie istnieje, zostanie wyświetlony komunikat o błędzie.
5. Komentarz: Sprawdź, czy plik raportu już istnieje. Wyjaśnienie: Ten komentarz informuje, że procedura sprawdza, czy plik raportu o podanej ścieżce już istnieje. Jeśli nie istnieje, zostanie utworzony nowy plik raportu.

6. Komentarz: Dołącz podane informacje do pliku raportu. Wyjaśnienie: Ten komentarz wskazuje, że procedura otwiera plik raportu w trybie dołączania i zapisuje podane informacje na końcu pliku.

Komentarze w kodzie są ważne, ponieważ pomagają innym programistom zrozumieć kod i jego działanie. Dzięki nim łatwiej jest zidentyfikować poszczególne części kodu i zrozumieć, co robią. Komentarze są również przydatne podczas późniejszych modyfikacji kodu, ponieważ ułatwiają zrozumienie intencji oryginalnego autora kodu.

Komentarz do kodu funkcji WartoscNieOtwPlik

```
Function WartoscNieOtwPlik(ByVal str_Sciezka As String, _
str_FolderPlik As String, str_Arkusz As String, str_Komorka As
String) As Variant
Dim str_Arg As String
WartoscNieOtwPlik = ""
' Sprawdź, czy ścieżka kończy się znakiem "\". Jeśli nie, dodaj go.
If Right(str_Sciezka, 1) <> "\" Then str_Sciezka = str_Sciezka
& "\"
' Sprawdź, czy istnieje plik w podanej ścieżce i folderze. Jeśli
nie, zakończ funkcję.
If Dir(str_Sciezka & "\" & str_FolderPlik) = "" Then Exit Function
' Utwórz argument dla funkcji ExecuteExcel4Macro.
str_Arg = "'" & str_Sciezka & "[" & str_FolderPlik & "]" & _
str_Arkusz & "'!" & Range(str_Komorka).Address(True, True, xlR1C1)
' Wykonaj funkcję ExecuteExcel4Macro i przypisz wynik do
zmiennej WartoscNieOtwPlik.
On Error Resume Next
WartoscNieOtwPlik = ExecuteExcel4Macro(str_Arg)
End Function
```

Wyjaśnienia

Funkcja `WartoscNieOtwPlik` służy do pobierania wartości z zamkniętego pliku Excel. Poniżej znajduje się wyjaśnienie dodanych komentarzy w kodzie:

1. `Sprawdź, czy ścieżka kończy się znakiem "\". Jeśli nie, dodaj go.` – Ten komentarz wskazuje, że kod sprawdza, czy ścieżka do folderu kończy się znakiem "". Jeśli nie, dodaje ten znak na końcu ścieżki. Jest to ważne, ponieważ w przeciwnym razie funkcja może nie działać poprawnie.
2. `Sprawdź, czy istnieje plik w podanej ścieżce i folderze. Jeśli nie, zakończ funkcję.` – Ten komentarz informuje, że kod sprawdza, czy plik o podanej nazwie istnieje w podanej ścieżce i folderze. Jeśli plik nie istnieje, funkcja zostanie zakończona i zwróci pustą wartość.
3. `Utwórz argument dla funkcji ExecuteExcel4Macro.` – Ten komentarz wskazuje, że kod tworzy argument dla funkcji `ExecuteExcel4Macro`, która jest używana do wykonania makra w zamkniętym pliku Excel. Argument ten zawiera informacje o ścieżce, nazwie pliku, arkuszu i komórce, z której ma zostać pobrana wartość.
4. `Wykonaj funkcję ExecuteExcel4Macro i przypisz wynik do zmiennej WartoscNieOtwPlik.` – Ten komentarz informuje, że kod wykonuje funkcję `ExecuteExcel4Macro` z podanym argumentem i przypisuje wynik do zmiennej `WartoscNieOtwPlik`. Ta zmienna będzie przechowywać wartość pobraną z zamkniętego pliku Excel.

Dzięki tym komentarzom kod jest bardziej czytelny i łatwiejszy do zrozumienia dla innych programistów.

Komentarz do kodu funkcji IsDatabaseOpen

Funkcja `IsDatabaseOpen` sprawdza, czy plik bazy danych o podanej ścieżce jest otwarty przez inny proces. Zwraca wartość True, jeśli plik jest otwarty, False w przeciwnym razie.

```
Function IsDatabaseOpen(databasePath As String) As Boolean
' Sprawdza, czy plik bazy danych o podanej ścieżce jest otwarty
przez inny proces.
' Zwraca True, jeśli plik jest otwarty, False w przeciwnym razie.
Dim fileNumber As Long
' Wyłącz obsługę błędów, aby zapobiec błędom wykonania, jeśli plik
jest niedostępny.
On Error Resume Next
' Próba otwarcia pliku do odczytu i zapisu z wyłącznym dostępem.
fileNumber = FreeFile
Open databasePath For Random Access Read Write Lock Read Write
As fileNumber
' Sprawdź, czy wystąpił błąd podczas otwierania pliku.
If Err.Number <> 0 Then
IsDatabaseOpen = True ' Plik jest otwarty przez inny proces.
Else
IsDatabaseOpen = False ' Plik nie jest otwarty.
Close fileNumber ' Zamknij plik.
End If
On Error GoTo 0 ' Przywróć normalne zarządzanie błędami.
End Function
```

Wyjaśnienia

Kod funkcji `IsDatabaseOpen` sprawdza, czy plik bazy danych jest otwarty przez inny proces. Aby to zrobić, najpierw próbuje otworzyć plik do odczytu i zapisu z wyłącznym dostępem. Jeśli podczas otwierania pliku wystąpi błąd, oznacza to, że plik jest już otwarty przez inny proces, dlatego funkcja zwraca wartość True. W przeciwnym razie, jeśli plik został pomyślnie otwarty, funkcja zwraca wartość False i zamyka plik.

Komentarze w kodzie zostały dodane, aby wyjaśnić działanie poszczególnych części kodu. Dzięki nim łatwiej jest zrozumieć, co robi dany fragment kodu i jakie są oczekiwane wyniki. Komentarze są ważne, ponieważ pomagają innym programistom zrozumieć kod i wprowadzać w nim zmiany bez ryzyka wprowadzenia błędów.

Komentarz do kodu funkcji GetFolderPath

Funkcja GetFolderPath() jest używana do zwracania ścieżki folderu nadrzędnego skoroszytu. Warto dodać komentarze w kodzie, aby łatwiej zrozumieć jego działanie i cel.

```
Function GetFolderPath() As String
' Zwraca ścieżkę folderu nadrzędnego skoroszytu.
' Użyj tej funkcji w komórce, aby pobrać ścieżkę folderu.
On Error Resume Next
' Wyłącz obsługę błędów, aby zapobiec błędom wykonania, jeśli
ścieżka folderu nie jest dostępna.
Dim workbookPath As String
' Pobierz pełną ścieżkę skoroszytu
workbookPath = ThisWorkbook.FullName
If Err.Number = 0 Then
' Wyodrębnij i zwróć ścieżkę folderu
GetFolderPath = Left(workbookPath, InStrRev(workbookPath, "\"))
Else
' Wystąpił błąd, zwróć komunikat o błędzie
GetFolderPath = "Błąd: Skoroszyt nie jest jeszcze zapisany lub
ścieżka jest niedostępna."
End If
On Error GoTo 0
' Przywróć normalne zarządzanie błędami
End Function
```

Wyjaśnienia

Kod funkcji GetFolderPath() jest odpowiedzialny za zwracanie ścieżki folderu nadrzędnego skoroszytu. Aby ułatwić zrozumienie kodu, dodano komentarze, które opisują poszczególne kroki i działania.

Pierwszy komentarz informuje, że funkcja zwraca ścieżkę folderu nadrzędnego skoroszytu i może być używana w komórce do pobrania tej ścieżki.

Następnie, za pomocą instrukcji "On Error Resume Next", wyłączono obsługę błędów, aby zapobiec błędom wykonania, jeśli ścieżka folderu nie jest dostępna. Jest to ważne, ponieważ jeśli skoroszyt nie jest jeszcze zapisany lub ścieżka jest niedostępna, może wystąpić błąd.

Następnie, zmienna "workbookPath" jest używana do przechowywania pełnej ścieżki skoroszytu za pomocą wyrażenia "ThisWorkbook.FullName".

Następnie, za pomocą instrukcji warunkowej "If Err.Number = 0", sprawdzane jest, czy wystąpił błąd. Jeśli nie ma żadnego błędu, używając funkcji "Left" i "InStrRev", wyodrębniana jest ścieżka folderu nadrzędnego i przypisywana do zmiennej "GetFolderPath".

Jeśli wystąpił błąd, funkcja zwraca komunikat o błędzie "Błąd: Skoroszyt nie jest jeszcze zapisany lub ścieżka jest niedostępna."

Na końcu, za pomocą instrukcji "On Error GoTo 0", przywracane jest normalne zarządzanie błędami.

Dzięki dodanym komentarzom, kod jest bardziej czytelny i łatwiejszy do zrozumienia dla innych programistów.

Komentarz do kodu procedury ClearDataArea_3

```
Sub ClearDataArea_3() ' Czyści obszar danych rozpoczynając od
komórki A7 w arkuszu "Pliki".
Dim lastRow As Long
Dim lastColumn As Long
' Znajdź ostatni użyty wiersz i kolumnę w arkuszu
lastRow = Pliki.Cells(Pliki.Rows.Count, "C").End(xlUp).Row
lastColumn =
Pliki.Cells(6, Pliki.Columns.Count).End(xlToLeft).Column
' Sprawdź, czy są dane do wyczyszczenia
If lastRow >= 7 And lastColumn >= 1 Then
Pliki.Range(Pliki.Cells(7, 1),
Pliki.Cells(lastRow, lastColumn)).ClearContents
Else
MsgBox "Brak danych poprzedniego stanu.", vbInformation,
"Czyszczenie danych"
End If
End Sub
```

Wyjaśnienia

W tym kodzie mamy procedurę o nazwie "ClearDataArea_3", która służy do czyszczenia obszaru danych w arkuszu "Pliki".

Komentarz w pierwszej linii kodu informuje nas, że ta procedura jest odpowiedzialna za czyszczenie obszaru danych rozpoczynając od komórki A7 w arkuszu "Pliki".

Następnie deklarowane są dwie zmienne: "lastRow" i "lastColumn", które będą przechowywać informacje o ostatnim używanym wierszu i kolumnie w arkuszu.

Kolejne komentarze wyjaśniają, jak znaleźć ostatni używany wiersz i kolumnę w arkuszu. Wykorzystywane są funkcje "End" i "xlUp" oraz "xlToLeft", które pozwalają nam znaleźć ostatni używany wiersz i kolumnę w odpowiednich kierunkach.

Następnie mamy warunek "If", który sprawdza, czy istnieją dane do wyczyszczenia. Jeśli ostatni używany wiersz jest większy lub równy 7 i ostatnia używana kolumna jest większa lub równa 1, to procedura kontynuuje czyszczenie danych. W przeciwnym razie wyświetlany jest komunikat o braku danych.

Na końcu mamy instrukcję "ClearContents", która czyszczenie zawartości obszaru danych w arkuszu "Pliki".

Ten kod jest przykładem procedury w VBA, która czyszczenie obszaru danych w arkuszu na podstawie określonych warunków. Komentarze w kodzie pomagają zrozumieć, co robią poszczególne linie kodu i jak działa ta procedura.

Komentarz do kodu funkcji FullNameToFileName

```
Function FullNameToFileName(ByVal sFullName As Variant) As String
' Konwertuje pełną ścieżkę pliku na samą nazwę pliku.
Dim k As Integer
Dim sTest As String
' Sprawdź, czy ciąg wejściowy zawiera "[" wskazujący na adres
komórki Excela.
If InStr(1, sFullName, "[") > 0 Then
k = InStr(1, sFullName, "[")
' Wyodrębnij tekst między "[" a "]".
sTest = Mid$(sFullName, k + 1, InStr(1, sFullName, "]") - k - 1)
Else
' Jeśli "[" nie zostanie znalezione, wyszukaj ostatni znak "\".
For k = Len(sFullName) To 1 Step -1
If Mid$(sFullName, k, 1) = "\" Then Exit For
Next k
' Wyodrębnij tekst po ostatnim znaku "\".
sTest = Mid$(sFullName, k + 1, Len(sFullName) - k)
End If
' Zwróć wyodrębnioną nazwę pliku.
FullNameToFileName = sTest
End Function
```

Wyjaśnienia

Ten kod VBA zawiera funkcję `FullNameToFileName`, która konwertuje pełną ścieżkę pliku na samą nazwę pliku. Komentarze w kodzie są używane do opisu poszczególnych kroków i zmiennych.

1. Na początku deklarowane są zmienne `k` i `sTest`. Zmienna `k` będzie przechowywać indeks znaku "[" lub "" w zależności od przypadku, a zmienna `sTest` będzie przechowywać wyodrębnioną nazwę pliku.
2. Następnie sprawdzane jest, czy ciąg wejściowy zawiera "[" za pomocą funkcji `InStr`. Jeśli tak, to oznacza, że mamy do czynienia z adresem komórki Excela. Zmienna `k` jest ustawiana na indeks znaku "[" za pomocą funkcji `InStr`, a następnie za pomocą funkcji `Mid$` wyodrębniany jest tekst między "[" a "]" i przypisywany do zmiennej `sTest`.
3. Jeśli ciąg wejściowy nie zawiera "[", to oznacza, że mamy do czynienia z pełną ścieżką pliku. W pętli `For` od końca ciągu sprawdzane są kolejne znaki, aż do znalezienia znaku "". Indeks tego znaku jest przypisywany do zmiennej `k`. Następnie za pomocą funkcji `Mid$` wyodrębniany jest tekst po ostatnim znaku "" i przypisywany do zmiennej `sTest`.
4. Na końcu funkcja zwraca wyodrębnioną nazwę pliku za pomocą instrukcji `FullNameToFileName = sTest`.

Ten kod jest przykładem funkcji VBA, która może być używana do konwersji pełnej ścieżki pliku na samą nazwę pliku. Komentarze w kodzie są ważne, ponieważ pomagają zrozumieć poszczególne kroki i działanie funkcji.