```
Private Sub Worksheet_Change(ByVal Target As Range)

 ' Диапазон, для которого будет срабатывать макрос
 Dim targetRange As Range
 Dim ITK As String
 Dim dataSheetName As String
 Dim rowChangedColNum As Integer
 Dim deptsStartLetter As String
 Dim deptsFinishLetter As String
 Dim deptsRowAdress As String
 Dim deptsRowRange As Range



 ITK = ThisWorkbook.Name
 dataSheetName = "info"
 checkCellRange = Workbooks(ITK).Worksheets(dataSheetName).Range("J7").Value
 rowChangedColNum = Workbooks(ITK).Worksheets(dataSheetName).Range("J8").Value
 deptsStartLetter = Workbooks(ITK).Worksheets(dataSheetName).Range("J9").Value
 deptsFinishLetter = Workbooks(ITK).Worksheets(dataSheetName).Range("J10").Value

 Set targetRange = Range(checkCellRange)

 ' Проверяем, является ли изменяемая ячейка в заданном диапазоне
 If Not Intersect(Target, targetRange) Is Nothing Then

   ' получаем номер строки изменяемой ячейки
   Dim rowNum As Long
   rowNum = Target.row
   ' Получаем диапазон ячеек для проверки распределения проб по отделам
   deptsRowAdress = deptsStartLetter & rowNum & ":" & deptsFinishLetter & rowNum
   'MsgBox deptsRowAdress
   Set deptsRowRange = Range(deptsRowAdress)
   ' Проходим по каждому столбцу в диапазоне
   For i = 1 To deptsRowRange.Columns.Count
     ' Проверяем, заполнен ли столбец цифрами
     If Application.WorksheetFunction.CountIf(deptsRowRange.Columns(i), ">0") Then
       Cells(rowNum, rowChangedColNum + i - 1).Value = 1
     ' This functionality is temporarly off cause its necesarry to keep in df previously marked depts
     ' Notwithstanding it has been marked wright or wrongly
     'Else
     '  Cells(rowNum, rowChangedColNum + i - 1).Value = ""
     End If
   Next i
   ' Вносим изменения в выбранную строку определенного столбца
   'Cells(rowNum, rowChangedColNum).Value = 1
   ' Пример: столбец 5  = E

 End If

End Sub
```

```
Option Explicit

Function GetCode(ByVal rangeToCheck As Range, _
ByVal codes As Range, ByVal dateValue As Date, _
ByVal sep As String) As String

  ' Проверяем, что диапазон и коды переданы корректно
  If rangeToCheck Is Nothing Then
    GetCode = ""
    Exit Function
  End If

  Dim cell As Range
  Dim codeString As String
  Dim i As Integer
  Dim year As String
  Dim month As String
  Dim monthYear As String
  Dim dateMinus As Integer
  Dim datePlus As Integer
  Dim ITK As String
  Dim dataSheetName As String
  Dim maxProbesNumber As Integer
  Dim errorResult As String



  ' Значение по умолчанию для строки кодов (сепаратором)
  codeString = sep
  ' Присваиваем дате пустую строку
  monthYear = ""
  ' Берем значения для проверки валидности дат поступления (чего?)
  ITK = ThisWorkbook.Name
  dataSheetName = "info"
  dateMinus = Workbooks(ITK).Worksheets(dataSheetName).Range("J3").Value
  datePlus = Workbooks(ITK).Worksheets(dataSheetName).Range("J4").Value
  maxProbesNumber = Workbooks(ITK).Worksheets(dataSheetName).Range("J5").Value
  errorResult = Workbooks(ITK).Worksheets(dataSheetName).Range("J6").Value

  ' Проходим по каждому столбцу в диапазоне
  For i = 1 To rangeToCheck.Columns.Count
    ' Проверяем, заполнен ли столбец цифрами и не равен ли 0
    If Application.WorksheetFunction.CountIf(rangeToCheck.Columns(i), ">0") > 0 And rangeToCheck.Columns(i) < max
ProbesNumber Then
      ' Добавляем код, соответствующий столбцу
      codeString = codeString & codes.Columns(i) & sep
    ElseIf Application.WorksheetFunction.CountIf(rangeToCheck.Columns(i), ">0") > 0 And rangeToCheck.Columns(i) >=
 maxProbesNumber Then
      MsgBox "Probes number cannot be too big (" & rangeToCheck.Columns(i) & "), i.e. more than <" & maxProbesNum
ber & "> !!!"
      GetCode = errorResult
      Exit Function
      'GoTo ErrorHandler
    ElseIf rangeToCheck.Columns(i) < 0 Then
```

```
      MsgBox "Probes number (" & rangeToCheck.Columns(i) & ") cannot be negative, i.e. less than <0> !!!"
      GetCode = errorResult
      Exit Function
      'GoTo ErrorHandler
    'Else
      'GetCode = errorResult
      'Exit Function
      'GoTo ErrorHandler
    End If
  Next i

  On Error GoTo ErrorHandler
    If dateValue > (Now() - dateMinus) And dateValue < (Now() + datePlus) Then
      ' Получаем дату в виде строки
      ' ДОБАВИТЬ ПРОВЕРКУ НА ДАТУ!!!
      year = Format(dateValue, "yy")
      month = Format(dateValue, "mm")
      monthYear = year & sep & month
    ElseIf dateValue > 100 Then
      MsgBox "Inputted date (" & dateValue & ") is Wrong! It has to be between <" & (Now() - dateMinus) & "> AND <" & (
Now() + datePlus) & ">"
      GetCode = errorResult
      Exit Function
    Else
      GetCode = errorResult
      Exit Function
    End If
    'Debug.Print monthYear
    ' Объединяем с датой
    codeString = codeString & monthYear
    'Debug.Print codeString
ErrorHandler:
  ' Возвращаем строку кодов
  GetCode = codeString


End Function



Function ExtractMonthYear(ByVal dateValue As Date, ByVal sep As String) As String

  ' Извлечение года из значения даты
  Dim year As String
  year = Format(dateValue, "yy")

  ' Извлечение месяца из значения даты
  Dim month As String
  month = Format(dateValue, "mm")

  ' Создание итоговой строки в формате ГГ-ММ
  ExtractMonthYear = year & sep & month

End Function
```

```
Function CheckIfNeedUpdate(ByVal rangeToCheck As Range) As Integer

 ' Проверяем, что диапазон передан корректно
 If rangeToCheck Is Nothing Then
   CheckIfNeedUpdate = 0
   Exit Function
 End If

 ' Проходим по каждому столбцу в диапазоне
 For i = 1 To rangeToCheck.Columns.Count
   ' Проверяем, заполнен ли столбец цифрами
   If Application.WorksheetFunction.CountIf(rangeToCheck.Columns(i), ">0") Then
     CheckIfNeedUpdate = 1
     Exit Function
   End If
 Next i

 CheckIfNeedUpdate = 0

End Function
```

```
Option Explicit

Public Function CellAddress(ByVal rowNumber As Long, _
                            ByVal colNumber As Long, _
                            ByVal removeDollar As Long) As String
    Dim address As String
    Dim errrorOut As String



    ' If removeDollar != 1 returns cell address WITH $ symbols (locked cell address)
    ' ElseIf removeDollar = 1 returns cell address WITHOUT $ symbols (UNlocked cell address)

    ' checks if numbers less than 1
    If rowNumber < 1 Then
        MsgBox "row number cannot be less than 1. it's current value is: <" & rowNumber & ">"
        CellAddress = errrorOut
        Exit Function
    End If
    If colNumber < 1 Then
        MsgBox "column number cannot be less than 1. it's current value is: <" & colNumber & ">"
        CellAddress = errrorOut
        Exit Function
    End If

    address = Cells(rowNumber, colNumber).address

    On Error GoTo ErrorHandler
    If removeDollar = 1 Then
        CellAddress = Replace(address, "$", "")
    Else
        CellAddress = address
    End If
    Exit Function

ErrorHandler:
    CellAddress = address
End Function
```

```
Option Explicit

Function OptimizeVBA(ByRef Status As Boolean)

    If Status = True Then
        Application.ScreenUpdating = False
        Application.Calculation = xlCalculationManual
        Application.DisplayAlerts = False
        Application.EnableEvents = False
        'DoCmd.SetWarnings = False
    Else
        Application.ScreenUpdating = True
        Application.Calculation = xlCalculationAutomatic
        Application.DisplayAlerts = True
        Application.EnableEvents = True
        'DoCmd.SetWarnings = True
    End If

    End Function
```

m12_lock_unlock_sheet - 1

```
Option Explicit

Function UnlockSheet(ByVal ws As Worksheet, _
                 ByVal UnlockSheetPassword As String) As Integer
   ' ...
   On Error GoTo ErrorHandler
   ws.Unprotect passWord:=UnlockSheetPassword
   UnlockSheet = 1
   Exit Function
ErrorHandler:
   UnlockSheet = 0
End Function

Function LockSheet(ByVal ws As Worksheet, _
               ByVal UnlockSheetPassword As String) As Integer
   On Error GoTo ErrorHandler
   ws.Protect DrawingObjects:=False, Contents:=True, Scenarios:= _
     False, AllowFormattingCells:=True, AllowFormattingColumns:=True, _
     AllowFormattingRows:=True, AllowFiltering:=True, passWord:=UnlockSheetPassword
   LockSheet = 1
   Exit Function
ErrorHandler:
   LockSheet = 0
End Function
```

```
Option Explicit

Sub GetExcelFilePath()
    'Create and set dialog box as variable
    Dim dialogBox As FileDialog
    Set dialogBox = Application.FileDialog(msoFileDialogOpen)

    'Do not allow multiple files to be selected
    dialogBox.AllowMultiSelect = False

    'Set the title of of the DialogBox
    dialogBox.Title = "Select a file"

    'Set the default folder to open
    dialogBox.InitialFileName = ActiveWorkbook.Path

    'Clear the dialog box filters
    dialogBox.Filters.Clear

    'Apply file filters - use ; to separate filters for the same name
    dialogBox.Filters.Add "Excel workbooks", "*.xlsx;*.xls;*.xlsm"

    'Show the dialog box and output full file name
    If dialogBox.Show = -1 Then
        'ActiveSheet.Range("filePath").Value = dialogBox.SelectedItems(1)
        ActiveCell.Value = dialogBox.SelectedItems(1)
    End If
End Sub
```

m3_send_data_button - 1

```
Option Explicit

Sub Кнопка1_Щелчок()

  Dim deptsNumberDetected As Integer
  Dim idOPPcolNumber As Integer
  Dim idDeptColNumber As Integer
  Dim codeOPPcolNumber As Integer
  Dim codeDeptColNumber As Integer
  Dim RGBredWarning As Integer
  Dim RGBgreenWarning As Integer
  Dim RGBblueWarning As Integer
  Dim ITK As String
  Dim dataSheetName As String
  Dim passWord As String
  Dim deptsMainSheetName As String
  Dim i As Integer
  Dim ColNumbersToCopy As Variant
  Dim ColNumbersToPaste As Variant
  Dim s As String
  Dim arraysFROMTOLengthsEqual As Boolean
  Dim arraysDEPTSLengthsEqual As Boolean
  Dim ArrayFromLen As Integer
  Dim ArrayToLen As Integer
  Dim fileNames As Variant
  Dim needRefreshDataColumnNumbers As Variant
  Dim probesAmountColumnNumbers As Variant
  Dim deptsNames As Variant
  Dim fileNamesLen As Integer
  Dim needRefreshDataLen As Integer
  Dim probesAmountLen As Integer
  Dim deptsNamesLen As Integer
  Dim currentFileName As String
  Dim currentNeedRefreshDataColumnNumber As Integer
  Dim currentProbesAmountColumnNumber As Integer
  Dim currentDeptName As String
  Dim filesSuccessfullyProcessed As Integer
  Dim errorText As String
  Dim errorLog As String
  Dim txtFilePath As String
  Dim delimeter As String
  Dim timeStart As String
  Dim codeErrorText As String




  ' ДОБАВИТЬ ЕЩЕ ТРИ АРРЭЯ ДЛЯ ИТЕРАЦИИ ПО НИМ (В СООТВЕТСВИИ С ОТДЕЛОМ) ВНУТРИ ТЕЛА ЦИКЛА FOR!!!
  ' fileNames, needRefreshDataColumnNumbers, probesAmountColumnNumbers
  ' Три переменных для текущего значения из соответствующего эррея
  ' currentFileName (String), currentNeedRefreshDataColumnNumber (Integer), currentProbesAmountColumnNumber
(Integer)
  ' Счетчик успешных итераций по файлам (удалось открыть и обновить / отправить в нем данные)
  ' filesSuccessfullyProcessed (Integer)
  ' errorLog is a full report string of the errors occerred during macros processing
  ' it's planned to record TimeStamp + errorLog in a log txt-file in the same directory as main file has been set...
  ' it's necessary to create function doesFileExits(filePath String) -> Boolean



  arraysFROMTOLengthsEqual = True
  arraysDEPTSLengthsEqual = True
  errorLog = ""
  timeStart = Format(DateTime.Now, "yyyy-MM-dd hh:mm:ss")
  delimeter = "-||-"

  ' MAIN INFO
  ITK = ThisWorkbook.Name
  dataSheetName = "info"
  deptsNumberDetected = Workbooks(ITK).Worksheets(dataSheetName).Range("J22").Value
  passWord = Workbooks(ITK).Worksheets(dataSheetName).Range("J11").Value
  deptsMainSheetName = Workbooks(ITK).Worksheets(dataSheetName).Range("J20").Value
  txtFilePath = Workbooks(ITK).Worksheets(dataSheetName).Range("J23").Value
  codeErrorText = Workbooks(ITK).Worksheets(dataSheetName).Range("J6").Value
  ' column numbers
  idOPPcolNumber = Workbooks(ITK).Worksheets(dataSheetName).Range("K15").Value
  idDeptColNumber = Workbooks(ITK).Worksheets(dataSheetName).Range("K16").Value
  codeOPPcolNumber = Workbooks(ITK).Worksheets(dataSheetName).Range("K17").Value
  codeDeptColNumber = Workbooks(ITK).Worksheets(dataSheetName).Range("K18").Value
  ' Deprecated function RangeToArray() plz, don't use it!!!
  'ColNumbersToCopy = RangeToArray(Workbooks(ITK).Worksheets(dataSheetName).Range("K12:U12"))
  ColNumbersToCopy = Application.Transpose(Workbooks(ITK).Worksheets(dataSheetName).Range("K12:U12"))
  ColNumbersToCopy = FilterArray(ColNumbersToCopy)
  ColNumbersToPaste = Application.Transpose(Workbooks(ITK).Worksheets(dataSheetName).Range("K13:U13"))
  ColNumbersToPaste = FilterArray(ColNumbersToPaste)
  ' Check whither arrays got lengths are equal?
  ArrayFromLen = ArrayLen(ColNumbersToCopy)
  ArrayToLen = ArrayLen(ColNumbersToPaste)
  If ArrayFromLen <> ArrayToLen Then
    arraysFROMTOLengthsEqual = False
    GoTo ErrorHandler
  End If
  ' RGB colors for warning in depts files if code is not equal for row
  RGBredWarning = Workbooks(ITK).Worksheets(dataSheetName).Range("J19").Value
  RGBgreenWarning = Workbooks(ITK).Worksheets(dataSheetName).Range("K19").Value
  RGBblueWarning = Workbooks(ITK).Worksheets(dataSheetName).Range("L19").Value
  ' Other arrays with info about departments
  fileNames = Application.Transpose(Workbooks(ITK).Worksheets(dataSheetName).Range("K4:U4"))
  fileNames = FilterArray(fileNames)
  needRefreshDataColumnNumbers = Application.Transpose(Workbooks(ITK).Worksheets(dataSheetName).Range("K
5:U5"))
  needRefreshDataColumnNumbers = FilterArray(needRefreshDataColumnNumbers)
  probesAmountColumnNumbers = Application.Transpose(Workbooks(ITK).Worksheets(dataSheetName).Range("K14:
U14"))
  probesAmountColumnNumbers = FilterArray(probesAmountColumnNumbers)
  deptsNames = Application.Transpose(Workbooks(ITK).Worksheets(dataSheetName).Range("K2:U2"))
  deptsNames = FilterArray(deptsNames)
  ' Check whither arrays got lengths are equal?
  fileNamesLen = ArrayLen(fileNames)
  needRefreshDataLen = ArrayLen(needRefreshDataColumnNumbers)
  probesAmountLen = ArrayLen(probesAmountColumnNumbers)
  deptsNamesLen = ArrayLen(deptsNames)
  If fileNamesLen <> needRefreshDataLen Or _
     fileNamesLen <> probesAmountLen Or _
     fileNamesLen <> deptsNamesLen Or _
     needRefreshDataLen <> probesAmountLen Or _
     needRefreshDataLen <> deptsNamesLen Or _
     probesAmountLen <> deptsNamesLen Then
    arraysDEPTSLengthsEqual = False
    GoTo ErrorHandler
  End If

  'MsgBox deptsNumberDetected & ", " & passWord & ", " & idOPPcolNumber & ", " & idDeptColNumber & ", " & codeO
PPcolNumber & ", " & codeDeptColNumber & ", " & deptsMainSheetName
  'MsgBox "RGB: (" & RGBredWarning & ", " & RGBgreenWarning & ", " & RGBblueWarning & ")"
  'MsgBox ColNumbersToCopy
  'MsgBox ColNumbersToPaste
  's = ""
  'For Each element In ColNumbersToCopy
  '      s = s & element & ", "
  '  Next element
  'MsgBox s

  For i = 0 To deptsNumberDetected - 1
    Dim resultStr As String
    ' Проходимся по всем множествам, относящимся к каждому отделу (...)
    ' Проверяем существует ли указанный файл, если не существует, то скипаем эту итерацию...
    ' Если файл существует (в противном случае), то вызываем функцию по отправке данных в отдел...
    currentFileName = fileNames(i)
    currentNeedRefreshDataColumnNumber = needRefreshDataColumnNumbers(i)
    currentProbesAmountColumnNumber = probesAmountColumnNumbers(i)
    currentDeptName = deptsNames(i)
    ' Check for file exists or not? If not GoTo IterErrorHandler (UPDATE FUNCTION DOESN'T WORK ON CURRENT ITERATI
ON)!
    ' USE IF-ELSE STATEMENT INSTEAD OF GOTO KEY WORD!!!
    If CheckFileExists(currentFileName) Then
        ' UPDATE (REFRESH) FUNCTION WORKS HERE!
        resultStr = FilterAndCopySpecificColumns(currentFileName, _
                                deptsMainSheetName, _
                                currentNeedRefreshDataColumnNumber, _
                                idOPPcolNumber, _
                                idDeptColNumber, _
                                ColNumbersToCopy, _
                                ColNumbersToPaste, _
                                codeOPPcolNumber, _
                                codeDeptColNumber, _
                                codeErrorText, _
                                passWord, _
                                RGBredWarning, _
                                RGBgreenWarning, _
```

```
                                    RGBblueWarning)

       MsgBox "Department name: <" & currentDeptName & ">, iteration number = <" & i & ">, filename = <" _
             & currentFileName & ">, need refresh data columns number = <" _
             & currentNeedRefreshDataColumnNumber & ">, probes amount for current department columns number = 
<" _
             & currentProbesAmountColumnNumber & ">"

       If resultStr <> "" Then
          errorLog = errorLog + resultStr + delimeter
       Else
          filesSuccessfullyProcessed = filesSuccessfullyProcessed + 1
       End If

    Else
       errorText = "File <" & currentFileName & "> doesn't exist."

       MsgBox errorText
       errorLog = errorLog + errorText + delimeter

    End If
  Next i

  s = "<" & filesSuccessfullyProcessed & "> of overall <" & deptsNumberDetected & "> files have been successfully pr
ocessed." & delimeter

  MsgBox s



ErrorHandler:
    If arraysFROMTOLengthsEqual <> True Then
       errorText = "Impossible to send data to departments cause of Col Numbers From (CopyFrom) Has <" _
                 & ArrayFromLen & "> elements. While one To (CopyTo) has <" & ArrayToLen _
                 & "> elements. Need to fix it on the sheet <" & dataSheetName & ">!"
       MsgBox errorText
       errorLog = errorLog + errorText + delimeter
    End If

    If arraysDEPTSLengthsEqual <> True Then
       errorText = "Impossible to send data to departments cause of Data Errors: " _
                 & "file names number provided = <" & fileNamesLen & ">. " _
                 & "departments names number provided = <" & deptsNamesLen & ">. " _
                 & "need refresh data columns number provided = <" & needRefreshDataLen & ">. " _
                 & "probes amount data columns number provided = <" & probesAmountLen & ">. "

       MsgBox errorText
       errorLog = errorLog + errorText + delimeter
    End If


  errorLog = timeStart + delimeter + errorLog + s + delimeter + Format(DateTime.Now, "yyyy-MM-dd hh:mm:ss")
  SaveStringToFile txtFilePath, errorLog
```

```
m3_send_data_button - 5


End Sub
```

m4_get_depts_number - 1

```
Option Explicit

Function GetDeptsNumber(диапазон1 As Range, диапазон2 As Range, диапазон3 As Range, диапазон4 As Range) A
s Integer

    Dim i As Long
    Dim count1 As Long, count2 As Long, count3 As Long, count4 As Long
    Dim continuousCount As Long

    ' Проверка количества ячеек в диапазонах
    'MsgBox диапазон1.Cells.Count & ", " & диапазон2.Cells.Count & ", " & диапазон3.Cells.Count & ", " & диапазон4.
Cells.Count
    If диапазон1.Cells.Count <> диапазон2.Cells.Count Or диапазон1.Cells.Count <> диапазон3.Cells.Count Or диапа
зон1.Cells.Count <> диапазон4.Cells.Count Then
        GetDeptsNumber = -1
        Exit Function
        'GoTo finish
    End If

    ' Подсчет ячеек, удовлетворяющих условию в каждом диапазоне
    For i = 1 To диапазон1.Cells.Count
        If диапазон1.Cells(i).Value > 0 Then count1 = count1 + 1
        If диапазон2.Cells(i).Value > 0 Then count2 = count2 + 1
        If диапазон3.Cells(i).Value > 0 Then count3 = count3 + 1
        If диапазон4.Cells(i).Value > 0 Then count4 = count4 + 1

        ' Подсчет непрерывных ячеек с самого начала
        If диапазон1.Cells(i).Value > 0 And диапазон2.Cells(i).Value > 0 And диапазон3.Cells(i).Value > 0 And диапазон
4.Cells(i).Value > 0 And _
           диапазон1.Cells(i).Value <> "" And диапазон2.Cells(i).Value <> "" And диапазон3.Cells(i).Value <> "" And диап
азон4.Cells(i).Value <> "" Then
            continuousCount = continuousCount + 1
        Else
            GetDeptsNumber = continuousCount
            Exit Function
            'continuousCount = 0
        End If
        'MsgBox count1 & ", " & count2 & ", " & count3 & ", " & count4
        'MsgBox continuousCount
    Next i

    ' Проверка равенства количества ячеек, удовлетворяющих условию
    If count1 = count2 And count1 = count3 And count1 = count4 Then
        GetDeptsNumber = continuousCount
    Else
        GetDeptsNumber = -1
    End If
'finish:
End Function
```

```
Option Explicit

Public Function RangeToArray(inputRange As Range) As Variant()
Dim size As Integer
Dim inputValue As Variant, outputArray() As Variant

    ' inputValue will either be an variant array for ranges with more than 1 cell
    ' or a single variant value for range will only 1 cell
    inputValue = inputRange

    On Error Resume Next
    size = UBound(inputValue)

    If Err.Number = 0 Then
        RangeToArray = inputValue
    Else
        On Error GoTo 0
        ReDim outputArray(1 To 1, 1 To 1)
        outputArray(1, 1) = inputValue
        RangeToArray = outputArray
    End If

    On Error GoTo 0

End Function

Public Function FilterArray(inputArray As Variant) As Variant()
Dim outputArray() As Variant
Dim i As Integer
Dim j As Integer
Dim elem As Variant


    ReDim outputArray(UBound(inputArray))
    j = 0

    For Each elem In inputArray
        If elem <> "" Then
            outputArray(j) = elem
            j = j + 1
        End If
    Next

    ReDim Preserve outputArray(j - 1)

    FilterArray = outputArray

End Function


Public Function ArrayLen(arr As Variant) As Integer
    ArrayLen = UBound(arr) - LBound(arr) + 1
End Function
```

```
Option Explicit

Public Function CheckFileExists(ByVal filePath As String) As Boolean

    CheckFileExists = Dir(filePath, vbNormal) <> ""

End Function
```

m7_append_str_to_file - 1

```
Option Explicit

Sub SaveStringToFile(ByVal filePath As String, ByVal textLine As String)

    Dim fullPath As String
    On Error GoTo ErrorHandler

    ' is absolute path?
    If InStr(filePath, ":\") > 0 Then
        ' does dir exist?
        If Dir(Left(filePath, InStr(filePath, "\") - 1), vbDirectory) = "" Then
            ' dir doesn't exist
            MsgBox "The next directory doesn't exist: " & Left(filePath, InStr(filePath, "\") - 1)
            Exit Sub
        End If
        fullPath = filePath ' using full path
    Else
        ' Not absolute path -> use current file dir
        fullPath = ThisWorkbook.Path & "\" & filePath
    End If

    ' Open txt file for appending new info from string
    Open fullPath For Append As #1
    Print #1, textLine
    Close #1

    Exit Sub

ErrorHandler:
    MsgBox "An error occured while trying to save data in txt-file: " & Err.Description

End Sub
```

```
Option Explicit

Public Function FilterAndCopySpecificColumns( _
   ByVal wbTargetPath As String, _
   ByVal wsTargetName As String, _
   ByVal filterColumn As Integer, _
   ByVal idColumnFROM As Integer, _
   ByVal idColumnTO As Integer, _
   ByVal copyColumns As Variant, _
   ByVal pasteColumns As Variant, _
   ByVal codeOPPcolNumber As Integer, _
   ByVal codeDeptColNumber As Integer, _
   ByVal codeErrorText As String, _
   ByVal passWord As String, _
   ByVal redRGB As Integer, _
   ByVal greenRGB As Integer, _
   ByVal blueRGB As Integer) As String

   Dim wbSource As Workbook
   Dim wbTarget As Workbook
   Dim canOpenWB As Boolean
   Dim targetWBIsOpened As Boolean
   Dim targetWSExists As Boolean
   Dim wsSource As Worksheet
   Dim wsTarget As Worksheet
   Dim outResult As String
   Dim idValue As Variant
   Dim author As String
   Dim lastRowSource As Long, lastRowTarget As Long
   Dim filterRow As Long, targetRow As Long
   Dim lastColumnNumSource As Integer
   Dim codeFROM As String
   Dim i As Integer
   Dim rowsClearContents() As Long
   Dim filteredRowsNumber As Long
   Dim j As Integer
   Dim k As Integer
   Dim nullFiltered As Boolean



   OptimizeVBA (True)
   'function to set all the things you want to set, but hate keying in

   Application.AutomationSecurity = msoAutomationSecurityForceDisable
   'this should stop those pesky enable prompts

   ' Variables default values
   canOpenWB = False
   targetWBIsOpened = False
   outResult = ""
   targetWSExists = False
   nullFiltered = False
```

```
    ' Function body
    ' Basic and aim workbooks
    Set wbSource = ThisWorkbook ' Current (basic) workbook
    ' Check if target wb is already opened
    author = WorkbookOpenedBy(wbTargetPath)
    'MsgBox author
    If author <> "" Then
        canOpenWB = True
        targetWBIsOpened = True
        outResult = author
        GoTo ErrorHandler
    End If
    On Error GoTo ErrorHandler
    Set wbTarget = Workbooks.Open(wbTargetPath)
    canOpenWB = True

    Set wsSource = wbSource.ActiveSheet
    Set wsTarget = wbTarget.Sheets(wsTargetName)
    ' Check if target sheet not exists
    If wsTarget Is Nothing Then
        wbTarget.Close
        GoTo ErrorHandler
    End If
    targetWSExists = True
    ' Close target workbook
    ' ##### FILTER AND SENDING DATA TO TARGET WB & WS

    ' last row after filter apply in source
    lastRowSource = wsSource.Cells(wsSource.Rows.Count, filterColumn).End(xlUp).row
    ' MsgBox "source WS after filter rows number is: <" & lastRowSource & ">"
    ' skip header
    filterRow = 2
    j = 0
    'ЗАМЕНИТЬ lastColumnNumSource ЕСЛИ УВЕЛИЧИТСЯ КОЛИЧЕСТВО СТОЛБЦОВ!!!
    lastColumnNumSource = 56
    ' Filter data in source...
    wsSource.Range(wsSource.Cells(1, 1), wsSource.Cells(lastRowSource, lastColumnNumSource)) _
    .AutoFilter Field:=filterColumn, Criteria1:="1", Operator:=xlFilterValues
    'filteredRowsNumber = wsSource.AutoFilter.Range.SpecialCells(xlCellTypeVisible).Count
    filteredRowsNumber = wsSource.AutoFilter.Range.SpecialCells(xlCellTypeVisible).Areas(1).Rows.Count
    If filteredRowsNumber = 1 Then
        nullFiltered = True
        GoTo ErrorHandler
    End If
    ReDim rowsClearContents(filteredRowsNumber)

    MsgBox "source WS rows number filtered: <" & filteredRowsNumber & ">"
    'MsgBox "Filter while processing: <" & wbTargetPath & ">"


    If UnlockSheet(wsTarget, passWord) = 1 Then
```

```
        Dim gg As Boolean
        'MsgBox "WS Target has been successfully unlocked"
    End If

    Do While filterRow <> lastRowSource
        codeFROM = wsSource.Cells(filterRow, codeOPPcolNumber).Value
        If wsSource.Cells(filterRow, filterColumn).Value <> "" And _
        codeFROM <> "" And codeFROM <> codeErrorText Then
            idValue = wsSource.Cells(filterRow, idColumnFROM).Value ' Получаем значение id
            ' Проверяем, существует ли строка с таким id в целевой книге
            lastRowTarget = wsTarget.Cells(wsTarget.Rows.Count, idColumnTO).End(xlUp).row
            targetRow = 2

            'MsgBox "current id = <" & idValue & ">..." & ", lab code = <" & codeFROM & ">" & ", IDs in target WS: <" & last
RowTarget & ">"

            Do While targetRow <= lastRowTarget
                If wsTarget.Cells(targetRow, idColumnTO).Value = idValue Then
                    rowsClearContents(j) = filterRow
                    j = j + 1
                    'MsgBox "id = <" & idValue & "> has been found in target WS!"
                    ' row with equal ID is in source WS!!!
                    If targetRow <= wsSource.Rows.Count Then
                        For i = LBound(copyColumns) To UBound(copyColumns)
                            ' Copy values to target sheet cells
                            'MsgBox "trying to send data, row N = <" & targetRow & ">, array index N = <" & i & ">, from column
N = <" _
                            '       & copyColumns(i) & ">, to column N = <" & pasteColumns(i) & ">, to cell address: <" & CellAddre
ss(targetRow, pasteColumns(i), 1) _
                            '       & ">, from cell address: <" & CellAddress(targetRow, copyColumns(i), 1) & ">..."

                            '& wsSource.Cells(targetRow, copyColumns(i)).Value
                            'wsTarget.Cells(targetRow, pasteColumns(i)) = wsSource.Cells(targetRow, copyColumns(i)).Value
                            'Range attr. will help (?)
                            ' Debug.Print
                            ' differs lab codes then fill it with RGB color
                            If pasteColumns(i) = codeDeptColNumber And _
                            wsTarget.Range(CellAddress(targetRow, pasteColumns(i), 1)).Value <> codeFROM Then
                                wsTarget.Range(CellAddress(targetRow, pasteColumns(i), 1)).Font.Color = RGB(redRGB, greenRGB,
 blueRGB)
                                wsTarget.Range(CellAddress(targetRow, pasteColumns(i), 1)).Font.Bold = True
                            ElseIf pasteColumns(i) <> codeDeptColNumber Then
                                wsTarget.Range(CellAddress(targetRow, pasteColumns(i), 1)) = wsSource.Range(CellAddress(filter
Row, copyColumns(i), 1)).Value
                            End If
                            'MsgBox "data successfully has been sent"
                        Next i
                    End If
                    Exit Do ' ID has been successfully found -> exit from while loop
                End If
            targetRow = targetRow + 1 ' next row iteration starting...
            Loop
        MsgBox "data with pre-existed ID in target WS has been successfully sent"
        ' ID hasn't found in target WS!!!
        If targetRow > lastRowTarget Then
            ' HERE I PROCEED...
            ' ID value need to copy into target WS!!!
            wsTarget.Range(CellAddress(lastRowTarget + 1, idColumnTO, 1)) = wsSource.Range(CellAddress(filterRow, idC
olumnFROM, 1)).Value
            For i = LBound(copyColumns) To UBound(copyColumns)
                'wsTarget.Cells(lastRowTarget + 1, pasteColumns(i)).Value = wsSource.Cells(filterRow, copyColumns(i)).Valu
e
                wsTarget.Range(CellAddress(lastRowTarget + 1, pasteColumns(i), 1)) = wsSource.Range(CellAddress(filterR
ow, copyColumns(i), 1)).Value
            Next i
        End If
        End If
        filterRow = filterRow + 1
      Loop


ErrorHandler:

      ' ##### END OF SENDING DATA!!!


      If nullFiltered = True Then
        wbTarget.Close SaveChanges:=False
        Application.AutomationSecurity = msoAutomationSecurityLow
        OptimizeVBA (False)
        wsSource.AutoFilterMode = False
        FilterAndCopySpecificColumns = outResult
        Exit Function
      ElseIf canOpenWB <> True Then
        outResult = outResult & Err.Description & " with <" & wbTargetPath & ">"
        Application.AutomationSecurity = msoAutomationSecurityLow
        OptimizeVBA (False)
        'wsSource.AutoFilterMode = False
        FilterAndCopySpecificColumns = outResult
        Exit Function
      ElseIf targetWSExists <> True And targetWBIsOpened <> True Then
        wbTarget.Close SaveChanges:=False
        outResult = outResult & "Sheet <" & wsTargetName & "> doesn't exist in workbook <" & wbTargetPath & ">"
        Application.AutomationSecurity = msoAutomationSecurityLow
        OptimizeVBA (False)
        wsSource.AutoFilterMode = False
        FilterAndCopySpecificColumns = outResult
        Exit Function
      ' somebody opened target wb
      ElseIf targetWBIsOpened <> False Then
        Application.AutomationSecurity = msoAutomationSecurityLow
        OptimizeVBA (False)
        'wsSource.AutoFilterMode = False
        FilterAndCopySpecificColumns = outResult
        Exit Function
      ElseIf Err.Description <> "" Then
```

```
        MsgBox Err.Description
        outResult = outResult & Err.Description & " with <" & wbTargetPath & ">"
        Application.AutomationSecurity = msoAutomationSecurityLow
        OptimizeVBA (False)
        wsSource.AutoFilterMode = False
        FilterAndCopySpecificColumns = outResult
        Exit Function
    End If

    If LockSheet(wsTarget, passWord) = 0 Then
        outResult = outResult & " ws target lock problem, "
    End If

    FilterAndCopySpecificColumns = outResult
    Application.AutomationSecurity = msoAutomationSecurityLow
    OptimizeVBA (False)
    wbTarget.RefreshAll
    wbTarget.Close SaveChanges:=True

    ' Clear marks from successfully copied rows
    For k = 0 To UBound(rowsClearContents)
        On Error GoTo arrayError
        If k <> 1 Then
            wsSource.Range(CellAddress(k, filterColumn, 1)).ClearContents
        End If
arrayError:
    Next k

    wsSource.AutoFilterMode = False

End Function
```

```
Option Explicit

Public Function WorkbookOpenedBy(workbookName As String) As String

   Dim wb As Workbook
   Dim wbOpened As Boolean
   'Dim users As Variant
   'Dim row As Integer
   Dim User1 As String
   Dim Date1 As String
   Dim status1 As String
   Dim f
   Dim i
   Dim x
   Dim inUseBy
   Dim tempfile
   Dim filename As String


    tempfile = Environ("TEMP") + "\tempfile" + CStr(Int(Rnd * 1000))

   f = FreeFile
   i = InStrRev(workbookName, "\")
   If (i > 0) Then
      filename = Mid(workbookName, 1, i) + "~$" + Mid(workbookName, 1 + i)
   Else
      filename = "~$" + workbookName
   End If

   On Error GoTo ErrorHandler
   Dim fso
   Set fso = CreateObject("Scripting.FileSystemObject")

   fso.CopyFile filename, tempfile

   Open tempfile For Binary Access Read As #f
   Input #f, x
   Close (f)
   inUseBy = Mid(x, 2, Asc(x))
   fso.Deletefile tempfile
   Set fso = Nothing

   WorkbookOpenedBy = "Книга: <" & workbookName & "> открыта пользователем: " & inUseBy
   Exit Function

ErrorHandler:
   WorkbookOpenedBy = ""
End Function
```

```
'@TestModule
'@Folder("Tests")



Option Explicit
Option Private Module

Private Assert As Object
Private Fakes As Object

'@ModuleInitialize
Private Sub ModuleInitialize()
    'this method runs once per module.
    Set Assert = CreateObject("Rubberduck.AssertClass")
    Set Fakes = CreateObject("Rubberduck.FakesProvider")
End Sub

'@ModuleCleanup
Private Sub ModuleCleanup()
    'this method runs once per module.
    Set Assert = Nothing
    Set Fakes = Nothing
End Sub

'@TestInitialize
Private Sub TestInitialize()
    'This method runs before every test in the module..
End Sub

'@TestCleanup
Private Sub TestCleanup()
    'this method runs after every test in the module.
End Sub
```