Поиск текста с определенным начертанием (жирный, курсив и т.д.), выделенного определенным шрифтом с помощью VBA

И вставка в текст HTML-тегов.

Преамбула

Да, мне нравится 2003 MS Word, и на мой взгляд, текстового редактора лучше пока еще никто не написал. Я делал все нижеследующее для себя, но возможно, материал будет полезен изучающим VBA. Особенно если у них возникают вопросы, как найти текст, выделенный жирным, курсивом, или там определенным шрифтом.
Так вот. Написал я в Word'е заметку про какой-нибудь скрипт, красиво оформил, код выделен Курьером, заголовки отцентрованы, важные места выделены жирным, пункты меню указаны курсивом, ну и так далее. Теперь нужно текст переложить в уютненький бложег, да так, чтобы сохранить форматирование вставить куда надо нужные теги. Встроенное сохранение в формате HTML вставляет много лишних тегов, добавлять нужные теги руками – лишняя работа, а я ленивый. Что же делать? У нас есть VBA, который все сам за нас сделает.

Поиск нужной информации в тексте

VBA умеет обращаться к внутренней функции поиска Office, а функция эта умеет искать текст не только по ключевому слову, шаблону, но и находить текст с определенным форматированием.

Найдем, например, первое вхождение текста, который выделен жирным:

```
Selection.Find.ClearFormatting 'Очистка параметров поиска
    With Selection.Find
        .Font.Bold = True 'найдем текст, выделенный жирным
        .Text = ""
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
Selection.Find.Execute 'запуск поиска
```

Этот фрагмент кода, будучи вставлен в исполнимый макрос найдет первое вхождение текста **выделенного жирным**

Найдем текст, выделенный определенным шрифтом, например `Courier New`

```
Selection.Find.ClearFormatting 'Очистка параметров поиска
    With Selection.Find
        .Font.NameAscii = "Courier New" 'найдем текст,
выделенный Курьером
        .Text = ""
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .Format = True
```

```
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
Selection.Find.Execute 'запуск поиска
```

Или текст, который расположен по центру:

```
Selection.Find.ClearFormatting 'Очистка параметров поиска
    With Selection.Find
          .ParagraphFormat.Alignment = wdAlignParagraphCenter
          .Text = ""
          .Replacement.Text = ""
          .Forward = True
          .Wrap = wdFindContinue
          .Format = True
          .MatchCase = False
          .MatchWholeWord = False
          .MatchWildcards = False
          .MatchSoundsLike = False
          .MatchAllWordForms = False
    End With
Selection.Find.Execute 'запуск поиска
```

Ищем нужное

Нашли, и вроде бы все просто – бери `Selecton.Text` и присваивай ему значение `"открывающий_тег"+Selection.Text+"закрывающий_тег"`, да не тут-то было. Во-первых, `Selection.Find.Execute` имеет дурацкое свойство искать не кусок текста от и до, а искать по абзацам, т.е.

```
Если этот параграф выделен Курьером
И следующий тоже
```

`Find.Excecute` выделит нам только первый абзац, для поиска остальных операцию придется повторять в цикле, но и это еще не все. Если мы сразу будем вставлять теги, то получится много лишних. Несмотря на то, что абзацы с одинаковым оформлением пусть даже и идут друг за другом, функция поиска будет рассматривать каждый абзац, как отдельный случай. Посему, все найденные данные стоит сначала собрать в массив(ы):

```
    Dim Start_s() As Long ' Массив, для хранения стартовых
                          ' позиций найденных блоков
    Dim End_s() As Long ' Массив, для хранения конечных позиций
                        ' найденных блоков
    Dim I As Long ' переменная для цикла

    ctr = 0 ' Количество найденных элементов
    shiftctr = 0 'Счетчик сдвигов (см. далее)

Selection.Find.Execute ' Выполняем первый поиск по ранее
установленным условиям
    If Not Selection.Find.Found Then Exit Sub ' Если ничего не
нашли - уходим
```

```
    Do While Selection.Find.Found 'ищем в цикле все остальное,
пока что-нибудь находится
        ctr = ctr + 1
        ReDim Preserve Start_s(ctr) 'переопределяем размерность
массива
        ReDim Preserve End_s(ctr) ' с сохранением ранее
установленных элементов (ключевое слово Preserve)
        Start_s(ctr) = Selection.Start 'сохраняем начальную
позицию найденного текста
        End_s(ctr) = Selection.End 'и конечную
        Selection.Find.Execute ' выполняем поиск в цикле
    Loop
```

Но, как я уже говорил, `find` ищет по абзацам, в результате в наших массивах окажется несколько идущих друг за другом подряд абзацев с одинаковым шрифтом. Конечное значение таких интервалов будет совпадать с началом следующего.
Чтобы не вставлять лишние теги, проанализируем это условие, и сдвинем массив на нужное количество элементов.

Процедура сдвига

Оформим отдельную процедуру, которая будет сдвигать массив на указанное количество элементов с указанного элемента массива:

```
Private Sub ShiftArr(ByRef arr As Variant, pos As Long, ctr As
Long)
    For I = 1 To ctr
        For J = pos To UBound(arr) - 1
            arr(J) = arr(J + 1)
        Next J
    Next I
End Sub
```

Удаление лишнего

Теперь, с помощью этой процедуры объединим следующие друг за другом интервалы .
Напоминаю условие: конечное значение таких интервалов будет совпадать с началом следующего

```
For I = 1 To ctr - 1
        Do While End_s(I) = Start_s(I + 1)
            ShiftArr Start_s, I + 1, 1
            ShiftArr End_s, I, 1
            shiftctr = shiftctr + 1
        Loop
Next I
```

Теперь переопределим размерность массивов, обрезав ненужные элементы (для того мы и сохраняли счетчик сдвигов в переменной `shiftctr`)
Примечание: Функция `UBound (массив)` возвращает количество его элементов, его размерность.

```
ReDim Preserve Start_s(UBound(Start_s) - shiftctr)
ReDim Preserve End_s(UBound(End_s) - shiftctr)
```

Все это я объединил в отдельную процедуру, куда передаю только сами теги, а параметры поиска устанавливаю перед ее выполнением.

Вставка тегов

Осталось только вставить теги. Но необходимо помнить, если мы в процессе работы что-нибудь добавим в текст, то нужно будет пересчитать "координаты" следующих символов. Если символы расположены после добавленных, то к их позиции приплюсуется длина добавленного текста.
В переменной TagLen будет содержаться длина открывающего и закрывающего тега, а переменная AllTagLen – накапливать суммарные длины всех вставленных тегов.

```
TagLen = Len(OpenTag) + Len(CloseTag)
    AllTagLen = TagLen
    For I = 1 To UBound(Start_s)
        Selection.Start = Start_s(I)
        Selection.End = End_s(I)
        Selection.Text = OpenTag + Selection.Text + CloseTag
        If I <> UBound(Start_s) Then
            Start_s(I + 1) = Start_s(I + 1) + AllTagLen
            End_s(I + 1) = End_s(I + 1) + AllTagLen
            AllTagLen = AllTagLen + TagLen
        End If
    Next I
End Sub
```

Конечно, макрос можно еще долго дорабатывать – например, сделать так, чтоб перед многострочным кодом вставлялись теги `pre`, для сохранения форматирования, сделать автоматическую вставку тегов `a href` (брать для них информацию из ссылок, заключенных в квадратные скобки), заменять специальные символы на последовательности HTML, и т.д. Но все это, во-первых, уж совсем специфично, а во-вторых, меня пока и так устраивает. Кое-что не влом и вручную подправить. Но может быть еще вернусь к теме.