This is a test 1 of <sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub><sub>B</sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub></sub>

This is a test 1 of <</sub>sub>Bsubscripts<su</sub>b>123 and Asuperscripts456.

<https://msdn.microsoft.com/en-us/library/microsoft.office.interop.word.find.execute.ASPX>

## Syntax

C#

[**VB**](https://msdn.microsoft.com/en-us/library/microsoft.office.interop.word.find.execute.ASPX?cs-save-lang=1&cs-lang=vb#code-snippet-1)

bool Execute(

ref Object FindText,

ref Object MatchCase,

ref Object MatchWholeWord,

ref Object MatchWildcards,

ref Object MatchSoundsLike,

ref Object MatchAllWordForms,

ref Object Forward,

ref Object Wrap,

ref Object Format,

ref Object ReplaceWith,

ref Object Replace,

ref Object MatchKashida,

ref Object MatchDiacritics,

ref Object MatchAlefHamza,

ref Object MatchControl

)

#### Parameters

*FindText*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. The text to be searched for. Use an empty string ("") to search for formatting only. You can search for special characters by specifying appropriate character codes. For example, "^p" corresponds to a paragraph mark and "^t" corresponds to a tab character.

*MatchCase*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** to specify that the find text be case-sensitive. Corresponds to the **Match case** check box in the **Find and Replace** dialog box (**Edit** menu).

*MatchWholeWord*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** to have the find operation locate only entire words, not text that's part of a larger word. Corresponds to the **Find whole words only** check box in the **Find and Replace** dialog box.

*MatchWildcards*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** to have the find text be a special search operator. Corresponds to the **Use wildcards** check box in the **Find and Replace** dialog box.

*MatchSoundsLike*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** to have the find operation locate words that sound similar to the find text. Corresponds to the **Sounds like** check box in the **Find and Replace** dialog box.

*MatchAllWordForms*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** to have the find operation locate all forms of the find text (for example, "sit" locates "sitting" and "sat"). Corresponds to the **Find all word forms** check box in the **Find and Replace** dialog box.

*Forward*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** to search forward (toward the end of the document).

*Wrap*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. Controls what happens if the search begins at a point other than the beginning of the document and the end of the document is reached (or vice versa if *Forward* is set to **False**). This argument also controls what happens if there's a selection or range and the search text isn't found in the selection or range.

Can be one of the following [WdFindWrap](https://msdn.microsoft.com/en-us/library/microsoft.office.interop.word.wdfindwrap.aspx) constants:

**wdFindAsk** After searching the selection or range, Microsoft Word displays a message asking whether to search the remainder of the document.

**wdFindContinue** The find operation continues if the beginning or end of the search range is reached.

**wdFindStop** The find operation ends if the beginning or end of the search range is reached.

*Format*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** to have the find operation locate formatting in addition to or instead of the find text.

*ReplaceWith*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. The replacement text. To delete the text specified by the *Find* argument, use an empty string (""). You specify special characters and advanced search criteria just as you do for the *Find* argument. To specify a graphic object or other non-text item as the replacement, move the item to the Clipboard and specify "^c" for *ReplaceWith*.

*Replace*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. Specifies how many replacements are to be made: one, all, or none. Can be any [WdReplace](https://msdn.microsoft.com/en-us/library/microsoft.office.interop.word.wdreplace.aspx) constant:

**wdReplaceAll**

**wdReplaceNone**

**wdReplaceOne**

*MatchKashida*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** if find operations match text with matching kashidas in an Arabic language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you’ve selected or installed.

*MatchDiacritics*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** if find operations match text with matching diacritics in a right-to-left language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you’ve selected or installed.

*MatchAlefHamza*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** if find operations match text with matching alef hamzas in an Arabic language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you’ve selected or installed.

*MatchControl*

Type: [System.Object](http://msdn2.microsoft.com/EN-US/library/e5kfa45b)

Optional **Object**. **True** if find operations match text with matching bidirectional control characters in a right-to-left language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you’ve selected or installed.

#### Return value

Type: [System.Boolean](http://msdn2.microsoft.com/EN-US/library/a28wyd50)

## Remarks

This method returns **True** if the find operation is successful.

If *MatchWildcards* is **True**, you can specify wildcard characters and other advanced search criteria for the *FindText* argument. For example, "\*(ing)" finds any word that ends in "ing."

To search for a symbol character, type a caret (^), a zero (0), and then the symbol's character code. For example, "^0151" corresponds to an em dash (— ).

Unless otherwise specified, replacement text inherits the formatting of the text it replaces in the document. For example, if you replace the string "abc" with "xyz," occurrences of "abc" with bold formatting are replaced with the string "xyz" with bold formatting.

Also, if *MatchCase* is **False**, occurrences of the search text that are uppercase will be replaced with an uppercase version of the replacement text regardless of the case of the search and replacement text. Using the previous example, occurrences of "ABC" are replaced with "XYZ."

Set myRange = ActiveDocument.Range(Start:=0, End:=0)

myRange.InsertAfter "Water = H20"

myRange.Characters(10).Font.Subscript = True

## C#: Searching a text in Word and getting the range of the result

http://stackoverflow.com/questions/4016960/c-searching-a-text-in-word-and-getting-the-range-of-the-result

range.Find.Execute(

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing);

while (range.Find.Found)

{

//Get selected index.

// Do as you please with range...

//Positions: range.Start... range.End

//search again

range.Find.Execute(

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing);

}

Gets a range from word using find method and format it.

//Parameter contains what you want to find.

\_wordApp.Selection.Find.Execute(title);

Word.Range range = \_wordApp.Selection.Range;

if (range.Text.Contains(title))

{

//gets desired range here it gets last character to make superscript in range

Word.Range temprange = \_document.Range(range.End - 1, range.End);

temprange.Select();

Word.Selection currentSelection = \_wordApp.Selection;

currentSelection.Font.Superscript = 1;

}

OR

range.Find.Execute returns true if found, and sets range to the found range:

var range = doc.Range();

while ( range.Find.Execute("xxx") )

Debug.Print( range.Text );

Note that range.Find.Execute will search the range after range if range is already a match for the Find conditions (after the first range.Find.Execute).

For example, this VBA Macro will find only the second "b" :

Sub Macro1()

ActiveDocument.Range.Text = "abba"

Dim r As Range

Set r = ActiveDocument.Range(1, 2) ' the first "b"

Debug.Print r.Start; r.End ' prints " 1 2 "

Debug.Print r.Find.Execute("b") ' prints "True"

Debug.Print r.Start; r.End ' prints " 2 3 "

Debug.Print r.Find.Execute("b") ' prints "False" (if r.Find.Wrap = wdFindStop)

Debug.Print r.Start; r.End ' prints " 2 3 "

End Sub

http://stackoverflow.com/questions/14215308/c-sharp-word-automation-find-and-replace-long-text-field

DocReplaceField(ref Word.Document objDoc, string Field, string Value)

{

object missing = System.Reflection.Missing.Value;

Word.Range range = objDoc.Content;

object findtext = Field;

object f = false;

object findreplacement = Value;

object findforward = false;

object findformat = true;

object findwrap = WdFindWrap.wdFindContinue;

object findmatchcase = false;

object findmatchwholeword = false;

object findmatchwildcards = false;

object findmatchsoundslike = false;

object findmatchallwordforms = false;

object findreplace = WdReplace.wdReplaceAll;

range.Find.Execute(

findtext,

findmatchcase,

findmatchwholeword,

findmatchwildcards,

findmatchsoundslike,

findmatchallwordforms,

findforward,

findwrap,

findformat,

findreplacement,

findreplace,

missing,

missing,

missing,

missing);

}

Word Automation: Detect if page break is necessary?

http://stackoverflow.com/questions/825254/word-automation-detect-if-page-break-is-necessary

// Generate page break

object pageBreak = WdBreakType.wdPageBreak;

wordApp.Selection.InsertBreak(ref pageBreak);

line = wordApp.Selection.Information(wdFirstCharacterLineNumber)

col = wordApp.Selection.Information(wdFirstCharacterColumnNumber)

This is a test subscript123 and a test superscript1234.

With Selection.Find.Font

.Superscript = False

.Subscript = True

End With

With Selection.Find

.Text = ""

.Replacement.Text = ""

.Forward = True

.Wrap = wdFindAsk

.Format = True

.MatchCase = False

.MatchWholeWord = False

.MatchWildcards = False

.MatchSoundsLike = False

.MatchAllWordForms = False

End With

Selection.Find.Execute

Application.Selection.Find.ClearFormatting();

if (Application.Selection.Find.Execute(ref findText,

ref missing, ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing))

{

Help word automation with c# urgent please

Jul 21 2009 2:22 PM

<http://www.c-sharpcorner.com/forums/help-word-automation-with-c-sharp-urgent-please>

Hello I'm trying to create a doc from a template I can access bookmarks an everything is ok except when there is a bookmark inside a textbox in the template, I can't acces that nor the bookmarks in the header and footer from c# the reason that a bookmark is in a textbox is because I need that text to not move from the page just stay there always, anyway here is the class I'm Using to access word

|  |
| --- |
| using System; using System.ComponentModel;  using Microsoft.Office.Interop.Word; using Microsoft.Office.Core; using Microsoft.Office.Tools.Word; using Microsoft.Office.Tools;  namespace Oficios {       public class CCWordApp  {  private Microsoft.Office.Interop.Word.ApplicationClass oWordApplic; // a reference to Word application  private Microsoft.Office.Interop.Word.Document oDoc; // a reference to the document    public CCWordApp()  {  // activate the interface with the COM object of Microsoft Word  oWordApplic = new Microsoft.Office.Interop.Word.ApplicationClass();  }   // Open a file (the file must exists) and activate it  public void Open(string strFileName)  {  object fileName = strFileName;  object readOnly = false;  object isVisible = true;  object missing = System.Reflection.Missing.Value;   oDoc = oWordApplic.Documents.Open(ref fileName, ref missing, ref readOnly,  ref missing, ref missing, ref missing, ref missing, ref missing, ref missing,  ref missing, ref missing, ref isVisible, ref missing, ref missing, ref missing, ref missing);    oDoc.Activate();  }    // Open a new document  public void Open()  {  object missing = System.Reflection.Missing.Value;  oDoc = oWordApplic.Documents.Add(ref missing, ref missing, ref missing, ref missing);   oDoc.Activate();  }      public void Quit()  {  object missing = System.Reflection.Missing.Value;  oWordApplic.Application.Quit(ref missing, ref missing, ref missing);  }   public void Save()  {  oDoc.Save();  }   public void SaveAs(string strFileName)  {  object missing = System.Reflection.Missing.Value;  object fileName = strFileName;   oDoc.SaveAs(ref fileName, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing,  ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing);  }   // Save the document in HTML format  public void SaveAsHtml(string strFileName)  {  object missing = System.Reflection.Missing.Value;  object fileName = strFileName;  object Format = (int)Microsoft.Office.Interop.Word.WdSaveFormat.wdFormatHTML;  oDoc.SaveAs(ref fileName, ref Format, ref missing, ref missing, ref missing, ref missing, ref missing,  ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing);  }     public void InsertText(string strText)  {  oWordApplic.Selection.TypeText(strText);  }   public void InsertImage(string nameImg)  {      }   public void InsertLineBreak()  {  oWordApplic.Selection.TypeParagraph();  }  public void InsertLineBreak(int nline)  {  for (int i = 0; i < nline; i++)  oWordApplic.Selection.TypeParagraph();  }   // Change the paragraph alignement  public void SetAlignment(string strType)  {  switch (strType)  {  case "Center":  oWordApplic.Selection.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;  break;  case "Left":  oWordApplic.Selection.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;  break;  case "Right":  oWordApplic.Selection.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;  break;  case "Justify":  oWordApplic.Selection.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphJustify;  break;  }   }    // if you use thif function to change the font you should call it again with   // no parameter in order to set the font without a particular format  public void SetFont(string strType)  {  switch (strType)  {  case "Bold":  oWordApplic.Selection.Font.Bold = 1;  break;  case "Italic":  oWordApplic.Selection.Font.Italic = 1;  break;  case "Underlined":  oWordApplic.Selection.Font.Subscript = 0;  break;  }   }   // disable all the style   public void SetFont()  {  oWordApplic.Selection.Font.Bold = 0;  oWordApplic.Selection.Font.Italic = 0;  oWordApplic.Selection.Font.Subscript = 0;   }   public void SetFontName(string strType)  {  oWordApplic.Selection.Font.Name = strType;   }   public void SetFontSize(int nSize)  {  oWordApplic.Selection.Font.Size = nSize;   }   public void InsertPagebreak()  {  // VB : Selection.InsertBreak Type:=wdPageBreak  object pBreak = (int)Microsoft.Office.Interop.Word.WdBreakType.wdPageBreak;  oWordApplic.Selection.InsertBreak(ref pBreak);  }   // Go to a predefined bookmark, if the bookmark doesn't exists the application will raise an error   public void GotoBookMark(string strBookMarkName)  {  // VB : Selection.GoTo What:=wdGoToBookmark, Name:="nome"  //if( oWordApplic.Selection.Bookmarks.Exists(strBookMarkName)){  object missing = System.Reflection.Missing.Value;      object Bookmark = (int)Microsoft.Office.Interop.Word.WdGoToItem.wdGoToBookmark;  object NameBookMark = strBookMarkName;  oWordApplic.Selection.GoTo(ref Bookmark, ref missing, ref missing, ref NameBookMark);  //oWordApplic.ActiveDocument.Bookmarks.Exists(strBookMarkName); // Range cosa = oWordApplic.ActiveDocument.Bookmarks.get\_Item(strBookMarkName).Range.Select();    // }  }   public void GotoFooter(string footer, string header, string strBookMarkName)  {  // VB : Selection.GoTo What:=wdGoToBookmark, Name:="nome"  //if( oWordApplic.Selection.Bookmarks.Exists(strBookMarkName)){  object missing = System.Reflection.Missing.Value;     object Bookmark2 = (int)Microsoft.Office.Interop.Word.WdGoToItem.wdGoToBookmark;  object NameBookMark = strBookMarkName;  // oWordApplic.Selection.GoTo(ref Bookmark, ref missing, ref missing, ref NameBookMark);  //oWordApplic.ActiveDocument.Bookmarks.Exists(strBookMarkName);  // Range cosa = oWordApplic.ActiveDocument.Bookmarks.get\_Item(strBookMarkName).Range.Select();   // }     Microsoft.Office.Interop.Word.Document dc = oWordApplic.ActiveDocument;  foreach (Microsoft.Office.Interop.Word.Section wordSection in dc.Sections)  {    object Bookmark = (int)Microsoft.Office.Interop.Word.WdGoToItem.wdGoToBookmark;  wordSection.Headers[Microsoft.Office.Interop.Word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.Text = header;  wordSection.Footers[Microsoft.Office.Interop.Word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.Text = footer;  wordSection.Headers[Microsoft.Office.Interop.Word.WdHeaderFooterIndex.wdHeaderFooterFirstPage].Range.Text = header;  wordSection.Footers[Microsoft.Office.Interop.Word.WdHeaderFooterIndex.wdHeaderFooterFirstPage].Range.Text = footer;    //wordSection.Headers[Microsoft.Office.Interop.Word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.Bookmarks.get\_Item(ref headersupder).Range.Text= header;  //wordSection.Headers[Microsoft.Office.Interop.Word.WdHeaderFooterIndex.wdHeaderFooterFirstPage].Range.Bookmarks.get\_Item(ref headersupder).Range.Text = header;  }        }   public void Try(string str)   {   object Name = str;  }   public void GoToTheEnd()  {  // VB : Selection.EndKey Unit:=wdStory  object missing = System.Reflection.Missing.Value;  object unit;  unit = Microsoft.Office.Interop.Word.WdUnits.wdStory;  oWordApplic.Selection.EndKey(ref unit, ref missing);   }  public void GoToTheBeginning()  {  // VB : Selection.HomeKey Unit:=wdStory  object missing = System.Reflection.Missing.Value;  object unit;  unit = Microsoft.Office.Interop.Word.WdUnits.wdStory;  oWordApplic.Selection.HomeKey(ref unit, ref missing);   }   public void GoToTheTable(int ntable)  {  // Selection.GoTo What:=wdGoToTable, Which:=wdGoToFirst, Count:=1, Name:=""  // Selection.Find.ClearFormatting  // With Selection.Find  // .Text = ""  // .Replacement.Text = ""  // .Forward = True  // .Wrap = wdFindContinue  // .Format = False  // .MatchCase = False  // .MatchWholeWord = False  // .MatchWildcards = False  // .MatchSoundsLike = False  // .MatchAllWordForms = False  // End With   object missing = System.Reflection.Missing.Value;  object what;  what = Microsoft.Office.Interop.Word.WdUnits.wdTable;  object which;  which = Microsoft.Office.Interop.Word.WdGoToDirection.wdGoToFirst;  object count;  count = 1;  oWordApplic.Selection.GoTo(ref what, ref which, ref count, ref missing);  oWordApplic.Selection.Find.ClearFormatting();   oWordApplic.Selection.Text = "";    }   public void GoToRightCell()  {  // Selection.MoveRight Unit:=wdCell   object missing = System.Reflection.Missing.Value;  object direction;  direction = Microsoft.Office.Interop.Word.WdUnits.wdCell;  oWordApplic.Selection.MoveRight(ref direction, ref missing, ref missing);  }   public void GoToLeftCell()  {  // Selection.MoveRight Unit:=wdCell   object missing = System.Reflection.Missing.Value;  object direction;  direction = Microsoft.Office.Interop.Word.WdUnits.wdCell;  oWordApplic.Selection.MoveLeft(ref direction, ref missing, ref missing);  }   public void GoToDownCell()  {  // Selection.MoveRight Unit:=wdCell   object missing = System.Reflection.Missing.Value;  object direction;  direction = Microsoft.Office.Interop.Word.WdUnits.wdLine;  oWordApplic.Selection.MoveDown(ref direction, ref missing, ref missing);  }   public void GoToUpCell()  {  // Selection.MoveRight Unit:=wdCell   object missing = System.Reflection.Missing.Value;  object direction;  direction = Microsoft.Office.Interop.Word.WdUnits.wdLine;  oWordApplic.Selection.MoveUp(ref direction, ref missing, ref missing);  }        } } |

using System;

using Spire.Doc;

using Spire.Doc.Documents;

using Spire.Doc.Fields;

namespace SubSuperScript

{

class Program

{

static void Main(string[] args)

{

Document document = new Document(@"Blank.doc");

Paragraph paragraph = document.LastSection.AddParagraph();

paragraph.AppendText("E = mc");

TextRange range1 = paragraph.AppendText("2");

//supperscript

range1.CharacterFormat.SubSuperScript = Spire.Doc.Documents.SubSuperScript.SuperScript;

paragraph.AppendBreak(BreakType.LineBreak);

paragraph.AppendText("F");

TextRange range2 = paragraph.AppendText("n");

//subscript

range2.CharacterFormat.SubSuperScript = Spire.Doc.Documents.SubSuperScript.SubScript;

paragraph.AppendText(" = F");

paragraph.AppendText("n-1").CharacterFormat.SubSuperScript =Spire.Doc.Documents.SubSuperScript.SubScript;

paragraph.AppendText(" + F");

paragraph.AppendText("n-2").CharacterFormat.SubSuperScript = Spire.Doc.Documents.SubSuperScript.SubScript;

//fontsize

foreach (var i in paragraph.Items)

{

if (i is TextRange)

{

(i as TextRange).CharacterFormat.FontSize = 36;

}

}

document.SaveToFile("result.docx", FileFormat.Docx);

System.Diagnostics.Process.Start("result.docx");

}

}

}