Desired layout & formatting:

```
407     bool TextSearch::IsDelimiter( const OUString& rStr, sal_Int32 nPos )
        const
408     {
409         bool bRet = 1;
410         if( '\x7f' != rStr[nPos])
411         {
412             if ( !xCharClass.is() )
413             {
414                 Reference < XInterface > xI = xMSF->createInstance(
415                         OUString::createFromAscii( "com.sun.star.i18n.Charact
        erClassification"));
416                 if( xI.is() )
417                     xI->queryInterface( ::getCppuType(
418                                 (const Reference< XCharacterClassification
        >*)0))
419                         >>= xCharClass;
420             }
421             if ( xCharClass.is() )
422             {
423                 sal_Int32 nCType = xCharClass->getCharacterType( rStr, nPos,
424                         aSrchPara.Locale );
425                 if( 0 != (( KCharacterType::DIGIT | KCharacterType::ALPHA |
426                             KCharacterType::LETTER ) & nCType ) )
427                     bRet = 0;
428             }
429         }
430         return bRet;
431     }
```

Currently possible with Writer:

With line numbering:

```
407 bool TextSearch::IsDelimiter( const OUString& rStr, sal_Int32 nPos )
408 const
409 {
410     bool bRet = 1;
411     if( '\x7f' != rStr[nPos])
412     {
413         if ( !xCharClass.is() )
414         {
415             Reference < XInterface > xI = xMSF->createInstance(
416                     OUString::createFromAscii( "com.sun.star.i18n.Charact
erClassification"));
417             if( xI.is() )
418                 xI->queryInterface( ::getCppuType(
419                             (const Reference< XCharacterClassification
>*)0))
420                     >>= xCharClass;
421         }
422         if ( xCharClass.is() )
423         {
424             sal_Int32 nCType = xCharClass->getCharacterType( rStr, nPos,
425                     aSrchPara.Locale );
426             if( 0 != (( KCharacterType::DIGIT | KCharacterType::ALPHA |
427                         KCharacterType::LETTER ) & nCType ) )
428                 bRet = 0;
429         }
430     }
431     return bRet;
432 }
```

Wrapped lines are numbered

With paragraph numbering:

```
407 bool TextSearch::IsDelimiter( const OUString& rStr, sal_Int32 nPos ) const
408 {
409     bool bRet = 1;
410     if( '\x7f' != rStr[nPos])
411     {
412         if ( !xCharClass.is() )
413         {
414             Reference < XInterface > xI = xMSF->createInstance(
415                     OUString::createFromAscii( "com.sun.star.i18n.CharacterClassificat
ion"));
416             if( xI.is() )
417                 xI->queryInterface( ::getCppuType(
418                             (const Reference< XCharacterClassification >*)0))
419                     >>= xCharClass;
420         }
421         if ( xCharClass.is() )
422         {
423             sal_Int32 nCType = xCharClass->getCharacterType( rStr, nPos,
424                     aSrchPara.Locale );
425             if( 0 != (( KCharacterType::DIGIT | KCharacterType::ALPHA |
426                         KCharacterType::LETTER ) & nCType ) )
427                 bRet = 0;
428         }
429     }
430     return bRet;
431 }
```

Background covers left indent
Right indent is lost

Sample source code with line numbering:

```
407     bool TextSearch::IsDelimiter( const OUString& rStr, sal_Int32 nPos )
408     const
409     {
410         bool bRet = 1;
411         if( '\x7f' != rStr[nPos])
412         {
413             if ( !xCharClass.is() )
414             {
415                 Reference < XInterface > xI = xMSF->createInstance(
416                         OUString::createFromAscii( "com.sun.star.i18n.Charact
417     erClassification"));
418                 if( xI.is() )
419                     xI->queryInterface( ::getCppuType(
420                                 (const Reference< XCharacterClassification
421     >*)0))
422                         >>= xCharClass;
423             }
424             if ( xCharClass.is() )
425             {
426                 sal_Int32 nCType = xCharClass->getCharacterType( rStr, nPos,
427                         aSrchPara.Locale );
428                 if( 0 != (( KCharacterType::DIGIT | KCharacterType::ALPHA |
429                                 KCharacterType::LETTER ) & nCType ) )
430                     bRet = 0;
431             }
432         }
433         return bRet;
434     }
```

This sample shows the desired layout for source code samples. The paragraph style provides the left and right indents and the gray background shading. The source lines that extend beyond the right paragraph margin are automatically wrapped by Writer; the wrapped lines are highlighted.

The line numbering, configured through Tools > Line Numbering, is not as desired: wrapped lines are counted, even though they are not distinct lines of source code. The code sample corresponds to lines 407–431; the line numbering does not match what the reader would see in a typical code display: TextSearch::IsDelimiter.

Sample code using paragraph numbering

Paragraph numbering can provide numbering that skips the wrapped lines. This works, but problems with the interaction of the numbering with the paragraph layout prevent it from providing the desired layout in this case.

Sample code without numbering:

```
    bool TextSearch::IsDelimiter( const OUString& rStr, sal_Int32 nPos )
    const
    {
        bool bRet = 1;
        if( '\x7f' != rStr[nPos])
        {
            if ( !xCharClass.is() )
            {
                Reference < XInterface > xI = xMSF->createInstance(
                        OUString::createFromAscii( "com.sun.star.i18n.Charact
    erClassification"));
                if( xI.is() )
                    xI->queryInterface( ::getCppuType(
                                (const Reference< XCharacterClassification
```

```
>*)0))
                    >>= xCharClass;
            }
            if ( xCharClass.is() )
            {
                sal_Int32 nCType = xCharClass->getCharacterType( rStr, nPos,
                        aSrchPara.Locale );
                if( 0 != (( KCharacterType::DIGIT | KCharacterType::ALPHA |
                                KCharacterType::LETTER ) & nCType ) )
                    bRet = 0;
            }
        }
        return bRet;
}
```

This sample uses a separate paragraph style with no line numbering, but otherwise the same layout as the first sample.

Now, we can add paragraph numbering by applying a list style.

Sample code with paragraph numbering:

```
407     bool TextSearch::IsDelimiter( const OUString& rStr, sal_Int32 nPos ) const
408     {
409         bool bRet = 1;
410         if( '\x7f' != rStr[nPos])
411         {
412             if ( !xCharClass.is() )
413             {
414                 Reference < XInterface > xI = xMSF->createInstance(
415                     OUString::createFromAscii( "com.sun.star.i18n.CharacterClassificat
    ion"));
416                 if( xI.is() )
417                     xI->queryInterface( ::getCppuType(
418                                 (const Reference< XCharacterClassification >*)0))
419                     >>= xCharClass;
420             }
421             if ( xCharClass.is() )
422             {
423                 sal_Int32 nCType = xCharClass->getCharacterType( rStr, nPos,
424                         aSrchPara.Locale );
425                 if( 0 != (( KCharacterType::DIGIT | KCharacterType::ALPHA |
426                                 KCharacterType::LETTER ) & nCType ) )
427                     bRet = 0;
428             }
429         }
430         return bRet;
431     }
```

With paragraph numbering, we can get the desired line numbering: there are no numbers for the wrapped lines. This is more appropriate for sample source listings and matches the usual source line numbering.

However, paragraph numbering is not entirely suitable for this purpose, primarily because the paragraph numbering interacts and interferes with the desired paragraph layout:

- The background color does not reflect the left margin of the source code paragraphs; instead, the background extends all the way to the numbers on the left.
- The right indent (“After text”) in the paragraph style is ignored altogether (this is Issue 109641).

Sample code with paragraph numbering and direct formatting:

```
407     bool TextSearch::IsDelimiter( const OUString& rStr, sal_Int32 nPos )
        const
408     {
409         bool bRet = 1;
410         if( '\x7f' != rStr[nPos])
411         {
412             if ( !xCharClass.is() )
413             {
414                 Reference < XInterface > xI = xMSF->createInstance(
415                         OUString::createFromAscii( "com.sun.star.i18n.Charact
        erClassification"));
416                 if( xI.is() )
417                     xI->queryInterface( ::getCppuType(
418                                 (const Reference< XCharacterClassification
        >*)0))
419                         >>= xCharClass;
420             }
421             if ( xCharClass.is() )
422             {
423                 sal_Int32 nCType = xCharClass->getCharacterType( rStr, nPos,
424                         aSrchPara.Locale );
425                 if( 0 != (( KCharacterType::DIGIT | KCharacterType::ALPHA |
426                                 KCharacterType::LETTER ) & nCType ) )
427                     bRet = 0;
428             }
429         }
430         return bRet;
431     }
```

Here, the right margin has been adjusted by direct formatting to match the right margin of first sample. This re-establishes the desired right indent and the line wrapping, but the paragraph background still doesn't match the target design: it extends out to the page margin.

This seems to be as close as possible to the desired numbering and layout.