N3062=10-0052
2010-3-10
Daveed Vandevoorde (daveed@edg.com)

# Core issue 789: Fixing Raw Strings wrt. Trigraphs (revision 1)

## Notes

In Summit, CWG agreed to deprecate trigraphs. In Frankfurt, IBM requested that an alternative solution be found that would not break their particular use of trigraphs in environments that assume EBCDIC encoding by default. This paper proposes wording to achieve that, while still addressing the spirit of comment UK 11 in N2800.

The gist of this proposal is to drop trigraphs altogether (instead of deprecating them) and replace their least uncommon uses by new tokens and new escape sequences (in non-raw strings). The "trigraph effect" is not emulated in raw strings and comments, nor are some perverse combinations like ??=# (for ##). I.e., this proposal attempts to address the major problem posed by trigraphs (that they counter-intuitively get substituted in raw strings), while enabling backward compatibility with any reasonable C++03 program in this respect. It would not be hard to add additional limitations on trigraph-like behavior: For example, trigraph-like substitution in strings could be disabled altogether, or it could be deprecated.

The changes are against N3035.

## Wording Changes

In 2.2 [lex.phases] paragraph 1 (in phase 1) delete the indicated sentence

> [...] Trigraph sequences (2.4) are replaced by corresponding single-character internal representations. [...]

and replace

> (i.e., using the \uXXXX notation)

by

> (e.g., using the \uXXXX notation)

In 2.2 [lex.phases] paragraph 1 replace (in phase 2)

> ... a backslash character (\) immediately followed ...

by

> ... a backslash character (\) or a character sequence **??/** immediately followed ...

In 2.3 [lex.charset] paragraph 1 add productions to the rule for *universal-character-name* as follows:

*universal-character-name:*
**\u** *hex-quad*
**??/u** *hex-quad*
**\U** *hex-quad hex-quad*
**??/U** *hex-quad hex-quad*

and in the text that follows replace

**\UNNNNNNNN**

by

**\UNNNNNNNN** or **??/UNNNNNNNN**

and

**\uNNNN**

by

**\uNNNN** or **??/uNNNN**

Delete subsection 2.4 [lex.trigraph].

In 2.6 [lex.digraph] replace footnote 15

15) These include “digraphs” and additional reserved words. The term “digraph” (token consisting of two characters) is not perfectly descriptive, since one of the alternative preprocessing-tokens is **%:%:** and of course several primary tokens contain two characters. Nonetheless, those alternative tokens that aren’t lexical keywords are colloquially known as “digraphs”.

by

15) The alternative tokens **<%**, **%>**, **<:**, **:>**, **%:**, and **%:%:** are colloquially known as “digraphs”. The alternative tokens containing a **??** character sequence are colloquially known as "trigraphs"; such character sequences were historically handled in translation phase 1 (2.2 _lex.phases_) but are now dealt with in later phases to allow for a more context-aware treatment.

and add entries to Table 2 as follows:

| Alternative | Primary | Alternative | Primary | Alternative | Primary |
|---|---|---|---|---|---|
| `<%` | `{` | `and` | `&&` | `and_eq` | `&=` |
| `%>` | `}` | `bitor` | `\|` | `or_eq` | `\|=` |
| `<:` | `[` | `or` | `\|\|` | `xor_eq` | `^=` |
| `:>` | `]` | `xor` | `^` | `not` | `!` |
| `%:` | `#` | `compl` | `~` | `not_eq` | `!=` |
| `%:%:` | `##` | `bitand` | `&` | `??'` | `^` |
| `??<` | `{` | `??!` | `\|` | `??'=` | `^=` |
| `??>` | `}` | `??!??!` | `\|\|` | `??!=` | `\|=` |
| `??(` | `[` | `??=` | `#` | `??-` | `~` |
| `??)` | `]` | `??=??=` | `##` | | |

Replace 2.9 [lex.header] paragraph 2

2 If either of the characters `'` or `\`, or either of the character sequences `/*` or `//` appears in a *q-char-sequence* or a *h-char-sequence*, or the character `"` appears in a *h-char-sequence*, the behavior is undefined.[18]

by

2 If either of the characters `'` or `\`, or any of the character sequences `??/`, `/*` or `//` appears in a *q-char-sequence* or a *h-char-sequence*, or the character `"` appears in a *h-char-sequence*, the behavior is undefined.[18]

In 2.13 [lex.operators] paragraph 1 add the following preprocessing tokens to the production for *preprocessing-op-or-punc*:

```
??<     ??>     ??(     ??)     ??=     ??=??=
??!     ??!??!  ??'     ??'=    ??!=    ??-
??/
```

In 2.14.3 [lex.ccon] add a production to the rule for *escape-sequence* as follows:

*escape-sequence:*
*simple-escape-sequence*
*octal-escape-sequence*
*hexadecimal-escape-sequence*
*trigraph-escape-sequence*

*trigraph-escape-sequence:* one of
`??=` `??'` `??(` `??)` `??!` `??<`
`??>` `??-` `??/n` `??/t` `??/v` `??/b`
`??/r` `??/f` `??/a` `??/??/` `??/?` `??/'`
`??/"`

and update the productions for *octal-escape-sequence* and *hexadecimal-escape-sequence* as follows:

*octal-escape-sequence:*
**** *octal-digit*
**** *octal-digit octal-digit*
**** *octal-digit octal-digit octal-digit*
**??/** *octal-digit*
**??/** *octal-digit octal-digit*
**??/** *octal-digit octal-digit octal-digit*

*hexadecimal-escape-sequence:*
**\x** *hexadecimal-digit*
**??/x** *hexadecimal-digit*
*hexadecimal-escape-sequence hexadecimal-digit*

In 2.14.3 [lex.ccon] replace paragraph 3

3 Certain nongraphic characters, the single quote `'`, the double quote `"`, the question mark `?`,[23] and the backslash `\`, can be represented according to Table 6. The double quote `"` and the question mark `?`, can be represented as themselves or by the escape sequences `\"` and `\?` respectively, but the single quote `'` and the backslash `\` shall be represented by the escape sequences `\'` and `\\` respectively. Escape sequences in which the character following the backslash is not listed in Table 6 are conditionally-supported, with implementation-defined semantics. An escape sequence specifies a single character.

by

3 Escape sequences specify a single (graphic or nongraphic) character as specified in Table 6. [ *Note:* The single quote **'** and the backslash **** characters must be represented using an escape sequence. Other characters from the basic source character set may be written directly or through an escape sequence. —*end note* ] Escape sequences consisting of a **** or **??/** followed by other characters but not listed in Table 6 are conditionally supported with implemented-defined semantics.

(thereby deleting footnote 23) and add entries to Table 6 as follows:

| | | |
|---|---|---|
| new-line | NL(LF) | **\n** or **??/n** |
| horizontal tab | HT | **\t** or **??/t** |
| vertical tab | VT | **\v** or **??/v** |
| backspace | BS | **\b** or **??/b** |
| carriage return | CR | **\r** or **??/r** |
| form feed | FF | **\f** or **??/f** |
| alert | BEL | **\a** or **??/a** |
| backslash | **** | **** or **??/??/** |
| question mark | **?** | **\?** or **??/?** |
| single quote | **'** | **\'** or **??/'** |
| double quote | **"** | **\"** or **??/"** |
| octal number | *ooo* | *****ooo* or **??/***ooo* |
| hex number | *hhh* | *****xooo* or **??/***xhhh* |
| octothorp | **#** | **??=** |
| circumflex | **^** | **??'** |
| left bracket | **[** | **??(** |
| right bracket | **]** | **??)** |
| vertical line | **\|** | **??!** |
| left brace | **{** | **??<** |
| right brace | **}** | **??>** |
| tilde | **~** | **??-** |

In 2.14.3 [lex.ccon] paragraph 4 replace the first two sentences

4 The escape `\`*ooo* consists of the backslash followed by one, two, or three octal digits that are taken to specify the value of the desired character. The escape `\`*xhhh* consists of the backslash followed by **`x`** followed by one or more hexadecimal digits that are taken to specify the value of the desired character.

by

4 The octal digits in an *octal-escape-sequence* are taken to specify the value of the desired character. The hexadecimal digits in a *hexadecimal-escape-sequence* are taken to specify the value of the desired character.

In the introductory grammar of 2.14.5 [lex.string] add to the rule for *raw-string*:

raw-string:
" *d-char-sequence*$_{opt}$ **`[`** *r-char-sequence*$_{opt}$ **`]`** *d-char-sequence*$_{opt}$ "
" *d-char-sequence*$_{opt}$ **`(`** *r-char-sequence*$_{opt}$ **`)`** *d-char-sequence*$_{opt}$ "

and to the rule for *d-char:*

*d-char*:
any member of the basic source character set except:
space, the left square bracket **`[`**, the right square bracket **`]`**,
the left parenthesis **`(`**, the right parenthesis **`)`**,
and the control characters representing horizontal tab,
vertical tab, form feed, and newline.

In 2.14.5 [lex.string] paragraph 14. replace the first sentence

14 Escape sequences in non-raw string literals and universal-character-names in string literals have the same meaning as in character literals (2.14.3), except that the single quote **`'`** is representable either by itself or by the escape sequence **`\'`**, and the double quote **`"`** shall be preceded by a **`\`**.

by

14 Escape sequences in non-raw string literals and universal-character-names in string literals have the same meaning as in character literals (2.14.3). [ Note: The double quote **`"`** and the backslash **`\`** characters must be represented using an escape sequence. Other characters from the basic source character set may be written directly or through an escape sequence. —*end note* ]