Chapter 53: Multi-Character Character Sequence

Section 53.1: Trigraphs

The symbols `[ ] { } ^ \ | ~ #` are frequently used in C programs, but in the late 1980s, there were code sets in use (ISO 646 variants, for example, in Scandinavian countries) where the ASCII character positions for these were used for national language variant characters (e.g. £ for # in the UK; `Æ Å æ å ø Ø` for `{ } { } | \` in Denmark; there was no `~` in EBCDIC). This meant that it was hard to write C code on machines that used these sets.

To solve this problem, the C standard suggested the use of combinations of three characters to produce a single character called a trigraph. A trigraph is a sequence of three characters, the first two of which are question marks.

The following is a simple example that uses trigraph sequences instead of #, { and }:

```
??=include <stdio.h>

int main()
??<
    printf("Hello World!\n");
??>
```

This will be changed by the C preprocessor by replacing the trigraphs with their single-character equivalents as if the code had been written:

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
}
```

Trigraph	Equivalent
??=	#
??/	\
??'	^
??(	[
??)	]
??!	|
??<	{
??>	}
??-	~

Note that trigraphs are problematic because, for example, `??/` is a backslash and can affect the meaning of continuation lines in comments, and have to be recognized inside strings and character literals (e.g. `'??/??/'` is a single character, a backslash).

Section 53.2: Digraphs

`Version ≥ C99`

In 1994 more readable alternatives to five of the trigraphs were supplied. These use only two characters and are known as digraphs. Unlike trigraphs, digraphs are tokens. If a digraph occurs in another token (e.g. string literals or

character constants) then it will not be treated as a digraph, but remain as it is.

The following shows the difference before and after processing the digraphs sequence.

```
#include <stdio.h>

int main()
<%
    printf("Hello %> World!\n"); /* Note that the string contains a digraph */
%>
```

Which will be treated the same as:

```
#include <stdio.h>

int main()
{
    printf("Hello %> World!\n"); /* Note the unchanged digraph within the string. */
}
```

Digraph Equivalent

<:	[
:>	]
<%	{
%>	}
%:	#