Improve the wording for Universal Character Names in identifiers

Document #:	P2620R2
Date:	2022-09-13
Programming Language C++	
Audience:	CWG

Reply-to: Corentin Jabot <corentin.jabot@gmail.com>

Abstract

Reword how *universal-character-names* in identifiers are specified.

Revisions

Revision 2

The previous revision of this paper proposed to allow UCNs outside of string literals to refer to basic character sets elements. However, we realized that we would not have a good model for UCNs in identifiers that are keywords. Neither allowing UCNs in keywords, or disallowing them seem desirable because of implementation burden and undue complexity.

It gets more challenging when considering macro names, contextual keywords and preprocessing directives. However, there was consensus that the other wording changes operated by this paper, ie moving the description of the UCNs handling in [lex.name] was desirable, and this paper now only contain that wording change. **This paper no longer contains any design change** and is retargeted at CWG.

Revision 1

- Fix typos
- Improve the wording by removing handling of UCNs from phase [lex.charset].

Wording

 Separate translation [lex.separate]

4. The source file is decomposed into preprocessing tokens and sequences of whitespace characters (including comments). A source file shall not end in a partial preprocessing token or in a partial comment. Each comment is replaced by one space character. New-line characters are retained. Whether each nonempty sequence of whitespace characters other than

new-line is retained or replaced by one space character is unspecified. ~~As characters from the source file are consumed to form the next preprocessing token (i.e., not being consumed as part of a comment or other forms of whitespace), except when matching a *c-char-sequence*, *s-char-sequence*, *r-char-sequence*, *h-char-sequence*, or *q-char-sequence*, *universal-character-name* s are recognized and replaced by the designated element of the translation character set.~~

The process of dividing a source file's characters into preprocessing tokens is context-dependent. [*Example:* See the handling of < within a `#include` preprocessing directive. — *end example*]

Character sets [lex.charset]

A *universal-character-name* designates the character in the translation character set whose UCS scalar value is the hexadecimal number represented by the sequence of *hexadecimal-digit* s in the *universal-character-name*. The program is ill-formed if that number is not a UCS scalar value. If a *universal-character-name* outside the *c-char-sequence*, *s-char-sequence*, or *r-char-sequence* of a *character-literal* or *string-literal* (in either case, including within a *user-defined-literal*) corresponds to a control character or to a character in the basic character set, the program is ill-formed. [*Note:* A sequence of characters resembling a *universal-character-name* in an *r-char-sequence* does not form a *universal-character-name*. — *end note*]

Identifiers [lex.name]

identifier:
identifier-start
identifier identifier-continue

identifier-start:
nondigit
an element of the translation character set of class XID_Start
universal-character-name
designating an element of the translation character set of class XID_*Start*

identifier-continue:
digit
nondigit
an element of the translation character set of class XID_Continue
universal-character-name
designating an element of the translation character set of class XID_*Continue*

nondigit: one of
`a b c d e f g h i j k l m`
`n o p q r s t u v w x y z`
`A B C D E F G H I J K L M`
`N O P Q R S T U V W X Y Z _`

digit: one of
`0 1 2 3 4 5 6 7 8 9`

The character classes XID_Start and XID_Continue are Derived Core Properties as described by UAX #44.

*universal-character-name*s are replaced by the designated element of the translation character set.

The program is ill-formed if an *identifier* does not conform to Normalization Form C as specified in ISO/IEC 10646. [*Note:* Identifiers are case-sensitive. — *end note*] [*Note:* In translation phase 4, *identifier* also includes those *preprocessing-token* s differentiated as keywords in the later translation phase 7. — *end note*]