#### Character and string literals.

Character and string literals are enclosed in quotes:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 | 'z'  'p'  "Hello world"  "How do you do?" |  |

The first two expressions represent *single-character literals*, and the following two represent *string literals* composed of several characters. Notice that to represent a single character, we enclose it between single quotes ('), and to express a string (which generally consists of more than one character), we enclose the characters between double quotes (").

Both single-character and string literals require quotation marks surrounding them to distinguish them from possible variable identifiers or reserved keywords. Notice the difference between these two expressions:

x

'x'

Here, x alone would refer to an identifier, such as the name of a variable or a compound type, whereas 'x' (enclosed within single quotation marks) would refer to the character literal 'x' (the character that represents a lowercase *x* letter).

Character and string literals can also represent special characters that are difficult or impossible to express otherwise in the source code of a program, like newline (\n) or tab (\t). These special characters are all of them preceded by a backslash character (\).

Here you have a list of the single character escape codes:

|  |  |
| --- | --- |
| **Escape code** | **Description** |
| \n | newline |
| \r | carriage return |
| \t | tab |
| \v | vertical tab |
| \b | backspace |
| \f | form feed (page feed) |
| \a | alert (beep) |
| \' | single quote (') |
| \" | double quote (") |
| \? | question mark (?) |
| \\ | backslash (\) |

For example:

'\n'  
'\t'  
"Left \t Right"  
"one\ntwo\nthree"

Internally, computers represent characters as numerical codes: most typically, they use one extension of the [ASCII](https://cplusplus.com/ascii) character encoding system (see [ASCII code](https://cplusplus.com/ascii) for more info). Characters can also be represented in literals using its numerical code by writing a backslash character (\) followed by the code expressed as an octal (base-8) or hexadecimal (base-16) number. For an octal value, the backslash is followed directly by the digits; while for hexadecimal, an *x* character is inserted between the backslash and the hexadecimal digits themselves (for example: \x20 or \x4A).

Several string literals can be concatenated to form a single string literal simply by separating them by one or more blank spaces, including tabs, newlines, and other valid blank characters. For example:

|  |  |  |
| --- | --- | --- |
| 1 2 | "this forms" "a single" " string "  "of characters" |  |

The above is a string literal equivalent to:

|  |  |  |
| --- | --- | --- |
|  | "this formsa single string of characters" |  |

Note how spaces within the quotes are part of the literal, while those outside them are not.

Some programmers also use a trick to include long string literals in multiple lines: In C++, a backslash (\) at the end of line is considered a *line-continuation* character that merges both that line and the next into a single line. Therefore, the following code:

|  |  |  |
| --- | --- | --- |
| 1 2 | x = "string expressed in \  two lines" |  |

is equivalent to:

|  |  |  |
| --- | --- | --- |
|  | x = "string expressed in two lines" |  |

All the character literals and string literals described above are made of characters of type char. A different character type can be specified by using one of the following prefixes:

|  |  |
| --- | --- |
| **Prefix** | **Character type** |
| u | char16\_t |
| U | char32\_t |
| L | wchar\_t |

Note that, unlike type suffixes for integer literals, these prefixes are *case sensitive*: lowercase for *char16\_t* and uppercase for *char32\_t* and *wchar\_t*.

For string literals, apart from the above u, U, and L, two additional prefixes exist:

|  |  |
| --- | --- |
| **Prefix** | **Description** |
| u8 | The string literal is encoded in the executable using UTF-8 |
| R | The string literal is a raw string |

In raw strings, backslashes and single and double quotes are all valid characters; the content of the literal is delimited by an initial R"*sequence*( and a final )*sequence*", where *sequence* is any sequence of characters (including an empty sequence). The content of the string is what lies inside the parenthesis, ignoring the delimiting sequence itself. For example:

|  |  |  |
| --- | --- | --- |
| 1 2 | R"(string with \backslash)"  R"&%$(string with \backslash)&%$" |  |

#### Both strings above are equivalent to "string with \\backslash". The R prefix can be combined with any other prefixes, such as u, L or u8.