std::strtok

Defined in header <cstring>

```
char* strtok( char* str, const char* delim );
```

Finds the next token in a null-terminated byte string pointed to by `str`. The separator characters are identified by null-terminated byte string pointed to by `delim`.

This function is designed to be called multiple times to obtain successive tokens from the same string.

- If `str` is not a null pointer, the call is treated as the first call to `strtok` for this particular string. The function searches for the first character which is *not* contained in `delim`.
  - If no such character was found, there are no tokens in `str` at all, and the function returns a null pointer.
  - If such character was found, it is the *beginning of the token*. The function then searches from that point on for the first character that *is* contained in `delim`.
    - If no such character was found, `str` has only one token, and the future calls to `strtok` will return a null pointer
    - If such character was found, it is *replaced* by the null character `'\0'` and the pointer to the following character is stored in a static location for subsequent invocations.
  - The function then returns the pointer to the beginning of the token
- If `str` is a null pointer, the call is treated as a subsequent call to `strtok`: the function continues from where it left in previous invocation. The behavior is the same as if the previously stored pointer is passed as `str`.

Parameters

str	–	pointer to the null-terminated byte string to tokenize
delim	–	pointer to the null-terminated byte string identifying delimiters

Return value

Pointer to the beginning of the next token or a `nullptr` if there are no more tokens.

Notes

This function is destructive: it writes the `'\0'` characters in the elements of the string `str`. In particular, a string literal cannot be used as the first argument of `std::strtok`.

Each call to this function modifies a static variable: is not thread safe.

Unlike most other tokenizers, the delimiters in `std::strtok` can be different for each subsequent token, and can even depend on the contents of the previous tokens.

Example

Run this code

```
#include <cstring>
#include <iostream>
#include <iomanip>

int main()
{
    char input[] = "one + two * (three - four)!";
    const char* delimiters = "! +- (*)";
    char *token = std::strtok(input, delimiters);
    while (token) {
        std::cout << std::quoted(token) << ' ';
        token = std::strtok(nullptr, delimiters);
    }

    std::cout << "\nContents of the input string now:\n\"";
    for (std::size_t n = 0; n < sizeof input; ++n) {
```

```
        if (const char c = input[n]; c != '\0')
            std::cout << c;
        else
            std::cout << "\\0";
    }
    std::cout << "\"\n";
}
```

Output:

```
"one" "two" "three" "four"
Contents of the input string now:
"one\0+ two\0* (three\0- four\0!\0"
```

See also

strpbrk	finds the first location of any character from a set of separators (function)
strcspn	returns the length of the maximum initial segment that consists of only the characters not found in another byte string (function)
strspn	returns the length of the maximum initial segment that consists of only the characters found in another byte string (function)
ranges::split_view **views::split** (C++20)	a view over the subranges obtained from splitting another view using a delimiter (class template) (range adaptor object)
C documentation for **strtok**	