Description

The strtok() function breaks the string s1 into tokens and null-terminates them. Delimiter-Characters at the beginning and end of str are skipped. On each subsequent call delim may change

The C90 prototype is:

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

The C99 prototype adds restrict-qualifiers:

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

Implementation

In standard C, this can be implemented as (adjust prototype for C99):

```
#include <string.h> /* strspn() strcspn() */
char *strtok(char * str, const char * delim)
{
    static char* p=0;
    if(str)
        p=str;
    else if(!p)
        return 0;
    str=p+strspn(p,delim);
    p=str+strcspn(str,delim);
    if(p==str)
        return p=0;
    p = *p ? *p=0,p+1 : 0;
    return str;
}
```

```
#include <cstring>
char *strtok( char *str1, const char *str2 );
```

The strtok() function returns a pointer to the next "token" in *str1*, where *str2* contains the delimiters that determine the token. strtok() returns **NULL** if no token is found. In order to convert a string to tokens, the first call to strtok() should have *str1* point to the string to be tokenized. All calls after this should have *str1* be **NULL**.

For example:

```
char str[] = "now # is the time for all # good men to
come to the # aid of their country";
char delims[] = "#";
char *result = NULL;
result = strtok( str, delims );
while( result != NULL ) {
  printf( "result is \"%s\"\n", result );
  result = strtok( NULL, delims );
}
```

The above code will display the following output:

```
result is "now "
result is " is the time for all "
result is " good men to come to the "
result is " aid of their country"
```

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.

If str != NULL, the function searches for the first character which is not separator. This character is the beginning of the token. Then the function searches for the first separator character. This character is the end of the token. Function terminates and returns NULL if end of str is encountered before end of the token is found. Otherwise, a pointer to end of the token is saved in a static location for subsequent invocations. This character is then replaced by a NULL-character and the function returns a pointer to the beginning of the token.

If str == NULL, 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 a token if the end of string has not been encountered. Otherwise returns NULL.

Note

The function is not thread safe.

Example

```
#include <cstring>

#include <iostream>

 int main()

{

    char input[100] = "A bird came down the walk";

    char *token = std::strtok(input, " ");

    while (token != NULL) {

        std::cout << token << '\n';

        token = std::strtok(NULL, " ");

    }

}
```

Output:

```
A
bird
came
down
the
walk
```