EXAMPLE 8.20 The `strpbrk()` Function

```
#include <cstring>
#include <iostream>
using namespace std;
int main()
{ char s[] = "The Mississippi is a long river.";
  cout << "s = \"" << s << "\"\n";
  char* p = strpbrk(s, "nopqr");
  cout << "strpbrk(s, \"nopqr\") points to s[" << p - s << "].\n";
  p = strpbrk(s, "NOPQR");
  if (p == NULL) cout << "strpbrk(s, \"NOPQR\") returns NULL.\n";
}
```

```
s = "The Mississippi is a long river."
strpbrk(s, "nopqr") points to s[12].
strpbrk(s, "NOPQR") returns NULL.
```

The call `strpbrk(s, "nopqr")` returns the first occurrence in `s` of any of the five characters `'n'`, `'o'`, `'p'`, `'q'`, or `'r'`. The first of these found is the `'p'` at `s[12]`.

The call `strpbrk(s, "NOPQR")` returns the `NULL` pointer because none of these five characters occurs in `s`.

The following table summarizes some of the most useful functions declared in `<cstring>`. Note that `size_t` is a special integer type that is defined in the `<cstring>` file.

`memcpy()`	`void* memcpy(void* s1, const void* s2, size_t n);` Replaces the first `n` bytes of `*s1` with the first `n` bytes of `*s2`. Returns `s`.
`strcat()`	`char* strcat(char* s1, const char* s2);` Appends `s2` to `s1`. Returns `s1`.
`strchr()`	`char* strchr(const char* s, int c);` Returns a pointer to the first occurrence of `c` in `s`. Returns `NULL` if `c` is not in `s`.
`strcmp()`	`int strcmp(const char* s1, const char* s2);` Compares `s1` with substring `s2`. Returns a negative integer, zero, or a positive integer, according to whether `s1` is lexicographically less than, equal to, or greater than `s2`.
`strcpy()`	`char* strcpy(char* s1, const char* s2);` Replaces `s1` with `s2`. Returns `s1`.
`strcspn()`	`size_t strcspn(char* s1, const char* s2);` Returns the length of the longest substring of `s1` that begins with `s1[0]` and contains none of the characters found in `s2`.
`strlen()`	`size_t strlen(const char* s);` Returns the length of `s`, which is the number of characters beginning with `s[0]` that precede the first occurrence of the `NUL` character.
`strncat()`	`char* strncat(char* s1, const char* s2, size_t n);` Appends the first `n` characters of `s2` to `s1`. Returns `s1`. If `n` ≥ `strlen(s2)`, then `strncat(s1,s2,n)` has the same effect as `strcat(s1,s2)`.