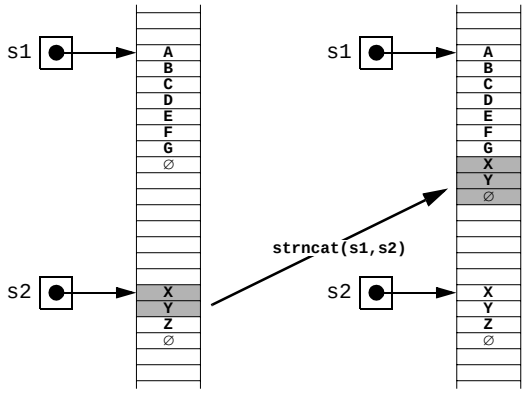
EXAMPLE 8.18 The Second String Concatenation Function `strncat()`

This program traces calls `strncat(s1,s2,n)`:

```
#include <cstring>
#include <iostream>
using namespace std;
int main()
{ // test-driver for the strncat() function:
  char s1[] = "ABCDEFG";
  char s2[] = "XYZ";
  cout << "Before strncat(s1,s2,2):\n";
  cout << "\ts1 = [" << s1 << "], length = " << strlen(s1) << endl;
  cout << "\ts2 = [" << s2 << "], length = " << strlen(s2) << endl;
  strncat(s1,s2,2);
  cout << "After strncat(s1,s2,2):\n";
  cout << "\ts1 = [" << s1 << "], length = " << strlen(s1) << endl;
  cout << "\ts2 = [" << s2 << "], length = " << strlen(s2) << endl;
}
```

```
Before strncat(s1,s2,2):
       s1 = [ABCDEFG], length = 7
       s2 = [XYZ], length = 3
After strncat(s1,s2,2):
       s1 = [ABCDEFGXY], length = 9
       s2 = [XYZ], length = 3
```

The call `strncat(s1,s2,2)` appends `XY` onto the end of `s1`. The effect can be visualized as shown here. Since `s2` has length 3, `strncat(s1,s2,2)` copies 2 bytes overwriting the `NUL` character of `s1` and the byte that follows it. Then it puts the `NUL` character in the next byte to complete the C-string `s1`. This increases its length to 9. (If either of the extra 2 bytes had been in use by some other object, then the entire 10 characters `ABCDEFGXYØ` would have been written in some other free part of memory.)



The next example illustrates the C-string *tokenize function*. Its purpose is to identify "tokens" within a given C-string: *e.g.*, words in a sentence.

EXAMPLE 8.19 The String Tokenize Function `strtok()`

This program shows how `strtok()` is used to extract the individual words from a sentence.

```
#include <cstring>
#include <iostream>
using namespace std;
int main()
{ // test-driver for the strtok() function:
  char s[] = "Today's date is March 12, 2000.";
  char* p;
  cout << "The string is: [" << s << "]\nIts tokens are:\n";
  p = strtok(s, " ");
```