EXAMPLE 8.18 The Second String Concatenation Function  \texttt{strncat()}  

This program traces calls  \texttt{strncat(s1,s2,n)}:  

\begin{verbatim}
#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):
    \hspace{1em} s1 = \texttt{[ABCDEFG]}, length = \texttt{7}\n    \hspace{1em} s2 = \texttt{[XYZ]}, length = \texttt{3}\n    \hspace{1em} strncat(s1,s2,2);
    \hspace{1em} cout << "After strncat(s1,s2,2):
    \hspace{1em} s1 = \texttt{[ABCDEFGXY]}, length = \texttt{9}\n    \hspace{1em} s2 = \texttt{[XYZ]}, length = \texttt{3}\n}
\end{verbatim}

The call  \texttt{strncat(s1,s2,2)}  appends  \texttt{XY}  onto the end of  \texttt{s1}. The effect can be visualized as shown here. Since  \texttt{s2}  has length 3,  \texttt{strncat(s1,s2,2)}  copies 2 bytes overwriting the  NUL  character of  \texttt{s1}  and the byte that follows it. Then it puts the  NUL  character in the next byte to complete the C-string  \texttt{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  \texttt{ABCDEFGXYØ}  would have been written in some other free part of memory.)

The next example illustrates the C-string  \texttt{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  \texttt{strtok()}  

This program shows how  \texttt{strtok()}  is used to extract the individual words from a sentence.  

\begin{verbatim}
#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: \texttt{[" << s << "]\n    \hspace{1em} Its tokens are: \texttt{\n    \hspace{1em} p = strtok(s, \texttt{" }});  
\end{verbatim}