A copy of the character at a particular position in a `string` variable may be obtained by using the member function:

```cpp
char at(int position);
// position:    position of desired element
```

For example:

```cpp
string s1 = "mairsy doates and doesy doates";
char ch1 = s1.at(5);   // ch1 == 'y'
```

Note that the positions in a string are numbered sequentially, starting at zero. So:

```cpp
for (int i = 7; i <= 12; i++)
    cout << s1.at(i) << ' ';
```

would print: `doates`
The character at a particular position in a `string` variable may also be referenced by using an index with the `string` object, similar to an array access.

For example:

```cpp
string s1 = "mairsy doates and doesy doates";
char ch1 = s1[5]; // ch1 == 'y'
```

The primary difference between `at()` and [] with `string` variables is that [] returns a reference to the `string` element, so:

```cpp
for (int i = 7; i <= 12; i++) {
    s1[i] = 'x';
    cout << s1[i] << ' ';
}
```

would print:  

```
x  x  x  x  x  x  x
```
A string of characters may be inserted at a particular position in a string variable by using the member function:

```cpp
string& insert(int startinsert, string s);
```

// startinsert: position at which insert begins
// s: string to be inserted

For example:

```cpp
string Name = "Fred Flintstone";
string MiddleInitial = " G."
Name.insert(4, MiddleInitial);
cout << Name << endl;
```

prints: Fred G. Flintstone

The function returns (a reference to) the string s1 which can be assigned to another string variable if desired; but the content of the original string is changed in any case.
Another version of the insert function takes four parameters:

\[
\text{string} & \text{ insert}(\text{int startinsert, string s, int startcopy, int numtocopy});
\]

// startinsert: position at which insert begins
// s: string to be inserted
// startcopy: position (in s) of first element to be used
// numtocopy: number of elements (of s) to be used

For example:

\[
\begin{align*}
\text{string s4} &= "0123456789"; \\
\text{string s5} &= "abcdefghijklmnopqrstuvwxyz"; \\
\text{s4.insert}(3, \text{s5, 7, 5}); \\
\text{cout} &<< "s4: " << \text{s4} << \text{endl}; \\
\end{align*}
\]

prints: s4: 012hijkl3456789

Note: a sequence of characters from a string is called a substring.
Extracting a Substring

A substring of a string may be extracted (copied) and assigned to another by using the member function:

```cpp
string& substr(int startcopy, int numtocopy);
```

// startcopy: position at which substring begins
// numtocopy: length of substring

For example:

```cpp
string s4 = "Fred Flintstone";
string s5 = s4.substr(5, 10);
cout << s4 << endl << s5 << endl;
```

prints: Fred Flintstone
   Flintstone
A substring may be deleted from a `string` by using the member function:

```cpp
string& erase(int starterase, int numtoerase);
```

// `starterase`: position of first element to be erased
// `numtoerase`: number of elements to be erased

For example:

```cpp
string s6 = "abcdefghijklmnopqrstuvwxyz";
s6.erase(3, 5);
cout << "s6: " << s6 << endl;
```

would print:  
`s6: abcijklmnopqrstuvwxyz`
A substring may be erased and replaced by another substring by using the member function:

```cpp
string& replace(int startreplace, int numtoreplace, string s);
```

// startreplace: position of first element to be replaced
// numtoreplace: number of elements to be replaced

For example:

```cpp
string s6 = "abcdefghijklmnopqrstuvwxyz";
string s7 = "Fred Flintstone";
s6.replace(3, 5, "01234");
s7.replace(0, 4, "Bradley");
cout << "s6: " << s6 << endl;
cout << "s7: " << s7 << endl;
```

would print:  
s6: abc01234ijklmnopqrstuvwxyz
s7: Bradley Flintstone
Searching for a Substring

A string may be searched for an occurrence of a substring by using the member function:

```cpp
int find(string s, int startsearch);
```

- `s`: substring to be searched for
- `startsearch`: position at which to begin search
- `returns`: position at which matching substring starts; string::npos if no match is found

For example:

```cpp
string s1 = "To be or not to be, that is the question.";
int loc = s1.find("be", 0);
int newloc = s1.find("be", loc + 1);
cout << loc << ' ' << newloc << endl;
```

prints: 3 16

Note: using loc instead of loc + 1 in the second call would result in finding the first occurrence again.
Putting several of the member functions together:

```cpp
string s1 = "But I have heard it works, even if you don't believe in it."

s1.erase(0, 4); // erase initial "But "

s1.replace(s1.find("even", 0), 4, "only"); // change "even" to "only"

s1.replace(s1.find("don't ", 0), 5, ""); // erase "don't " by replacing it
    //     with the empty string

cout << s1 << endl;
```

prints: I have heard it works, only if you believe in it.
This chapter includes only a minimal introduction to the world of string objects in C++.

There are many additional member functions. For example, there are ten different `replace` member functions in the standard C++ string library.

The interested reader is referred to Bjarne Stroustrup’s excellent *The C++ Programming Language, 3rd Ed.* for further details.