# Class string and String Stream Processing

# 18.1 Introduction (Cont.)

- `string` object
  - Initialization

```
string empty();
```

  - Creates an empty `string` containing no characters

```
string text( "hello" );
```

  - Creates a `string` containing the characters `"hello"`

```
string name( 8, 'x' );
```

  - Creates a `string` containing eight `'x'` characters

```
string month = "March";
```

  - Implicitly performs `string month( "March" );`

# 18.1 Introduction (Cont.)

- `string` object (Cont.)
  - No conversion from `int` or `char` in a `string` definition
    - Examples (produce syntax errors)

```
string error1 = 'c';
string error2( 'u' );
string error3 = 22;
string error4( 8 );
```

  - Assigning a single character to a `string` object is allowed
    - Example

```
string1 = 'n';
```

# 18.1 Introduction (Cont.)

- `string` object (Cont.)
  - Member functions `length`
    - Return the length of the `string`
  - The subscript operator `[]`
    - Used to access and modify individual characters
    - First subscript is `0`, last subscript is `length() - 1`

# 18.1 Introduction (Cont.)

- `string` object (Cont.)
  - Stream extraction operator (>>)
    - Example

```
cin >> stringObject;
```

    - Input is delimited by white-space characters
  - Function `getline` is overloaded for `strings`
    - Example

```
getline( cin, string1 );
```

    - Input is delimited by a newline ('\n');

# 18.2 string Assignment and Concatenation

- Member function `assign`
  - Copies the contents of a `string` into another `string`
  - Single-argument version
    - Copies contents of the `string` argument into the current `string`
  - Three-argument version
    - Copies a specified range of characters
    - Example
      - `targetString.assign( sourceString, start, numberOfCharacters );`

# 18.2 `string` Assignment and Concatenation (Cont.)

- **Member function `at`**
  - **Allows access to individual characters**
    - **Much like the subscript operator does**
  - **Provides checked access (or range checking)**
    - **Going past the end of the `string` throws an `out_of_range` exception**
    - **Subscript operator does not provide checked access**

# 18.2 `string` Assignment and Concatenation (Cont.)

- `string` concatenation
  - Member function `append`
    - **Single-argument version**
      - Concatenates contents of the `string` argument to end of the current `string`
    - **Three-argument version**
      - Concatenates specified range of characters from the `string` argument to end of the current `string`

Outline

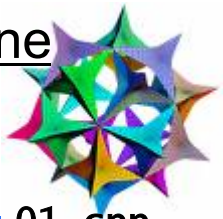

Fig18_01.cpp

(1 of 2)

```
1  // Fig. 18.1: Fig18_01.cpp
2  // Demonstrating string assignment and concatenation.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <string>
8  using std::string;
9
10 int main()
11 {
12    string string1( "cat" );
13    string string2;
14    string string3;
15
16    string2 = string1; // assign string1 to string2
17    string3.assign( string1 ); // assign string1 to string3
18    cout << "string1: " << string1 << "\nstring2: " << string2
19       << "\nstring3: " << string3 << "\n\n";
20
21    // modify string2 and string3
22    string2[ 0 ] = string3[ 2 ] = 'r';
23
24    cout << "After modification of string2 and string3:\n"
25       << string1 << "\nstring2: " << string2 << "\nstring3: ";
26
27    // demonstrating member function at
28    for ( int i = 0; i < string3.length(); i++ )
29       cout << string3.at( i );
```

Assign the value of **string1** to **string2** with the assignment operator

```
string1: cat
string2: cat
string3: cat
```

Copy **string1** into **string3** with the **assign** member function

Use the subscript operator to assign to individual characters

Use member functions **length** and **at** to output the contents of **string3** one character at a time

```
After modification of string2 and string3:
string1: cat
string2: rat
string3: car
```

```
30
31    // declare string4 and string5
32    string string4( string1 + "apult" ); // concatenation
33    string string5;
34
35    // overloaded +=
36    string3 += "pet"; // create "carpet"
37    string1.append( "acomb" ); // create "catacomb"
38
39    // append subscript locations 4 through end of string1 to
40    // create string "comb" (string5 was initially empty)
41    string5.append( string1, 4, string1.length() - 4 );
42
43    cout << "\n\nAfter concatenation:\nstring1: " << string1
44       << "\nstring2: " << string2 << "\nstring3: " << string3
45       << "\nstring4: " << string4 << "\nstring5: " << string5 << endl;
46    return 0;
47 } // end main
```

Initialize **string4** to the result of concatenating **string1** and **"apult"** using the addition operator **+**

Concatenate **string3** and **"pet"** using the addition assignment operator **+=**

(2 of 2)

Concatenate **string1** and **"acomb"**

Append the string **"comb"** (the characters from subscript **4** to the end of **string1**) to empty **string string5**

```
After concatenation:
string1: catacomb
string2: rat
string3: carpet
string4: catapult
string5: comb
```

# 18.3 Comparing strings

- Overloaded comparison operators
  - Operators ==, !=, <, >, <=, >= are overloaded for strings
    - All return bool values
- Member function compare
  - Compares the values of two strings
    - Returns 0 if the strings are equivalent
    - Returns positive number if the current string is lexicographically greater than the argument string
    - Returns negative number if the current string is lexicographically less than the argument string

# 18.3 Comparing strings (Cont.)

- **Member function `compare` (Cont.)**
  - **Overloaded versions**
    - **With five arguments**
      - First two arguments specify starting subscript and length in the current `string`
      - Third argument specifies the comparison `string`
      - Last two arguments specify starting subscript and length in the comparison `string`
    - **With three arguments**
      - First two arguments specify starting subscript and length in the current `string`
      - Third argument specifies the comparison `string`

```
1  // Fig. 18.2: Fig18_02.cpp
2  // Demonstrating string comparison capabilities.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <string>
8  using std::string;
9
10 int main()
11 {
12    string string1( "Testing the comparison functions." );
13    string string2( "Hello" );
14    string string3( "stinger" );
15    string string4( string2 );
16
17    cout << "string1: " << string1 << "\nstring2: " << string2
18       << "\nstring3: " << string3 << "\nstring4: " << string4 << "\n\n";
19
20    // comparing string1 and string4
21    if ( string1 == string4 )
22       cout << "string1 == string4\n";
23    else // string1 != string4
24    {
25       if ( string1 > string4 )
26          cout << "string1 > string4\n";
27       else // string1 < string4
28          cout << "string1 < string4\n";
29    } // end else
```

Outline

Fig18_02.cpp

(1 of 4)

Test **string1** against **string4** for equality using the overloaded equality operator

Test **string1** against **string4** using the overloaded greater-than operator

```
30
31    // comparing string1 and string2
32    int result = string1.compare( string2 );
33
34    if ( result == 0 )
35       cout << "string1.compare( string2 ) == 0\n";
36    else // result != 0
37    {
38       if ( result > 0 )
39          cout << "string1.compare( string2 ) > 0\n";
40       else // result < 0
41          cout << "string1.compare( string2 ) < 0\n";
42    } // end else
43
44    // comparing string1 (elements 2-5) and string3 (elemen
45    result = string1.compare( 2, 5, string3, 0, 5 );
46
47    if ( result == 0 )
48       cout << "string1.compare( 2, 5, string3, 0, 5 ) == 0\n";
49    else // result != 0
50    {
51       if ( result > 0 )
52          cout << "string1.compare( 2, 5, string3, 0, 5 ) > 0\n";
53       else // result < 0
54          cout << "string1.compare( 2, 5, string3, 0, 5 ) < 0\n";
55    } // end else
```

Compare **string1** to **string2**

Compare **"sting"** (from **string1**) to **"sting"** (from **string3**)

Outline

Fig18_02.cpp

(2 of 4)

```
56
57    // comparing string2 and string4
58    result = string4.compare( 0, string2.length(), string2 );
59
60    if ( result == 0 )
61       cout << "string4.compare( 0, string2.length(), "
62          << "string2 ) == 0" << endl;
63    else // result != 0
64    {
65       if ( result > 0 )
66          cout << "string4.compare( 0, string2.length(), "
67             << "string2 ) > 0" << endl;
68       else // result < 0
69          cout << "string4.compare( 0, string2.length(), "
70             << "string2 ) < 0" << endl;
71    } // end else
```

Compare "**Hello**" (from **string4**) to **string2**



```
72
73    // comparing string2 and string4
74    result = string2.compare( 0, 3, string4 );
75
76    if ( result == 0 )
77       cout << "string2.compare( 0, 3, string4 ) == 0" << endl;
78    else // result != 0
79    {
80       if ( result > 0 )
81          cout << "string2.compare( 0, 3, string4 ) > 0" << endl;
82       else // result < 0
83          cout << "string2.compare( 0, 3, string4 ) < 0" << endl;
84    } // end else
85
86    return 0;
87 } // end main
```

Compare "**Hel**" (from **string2**) to **string4**



Fig18_02.cpp



```
string1: Testing the comparison functions.
string2: Hello
string3: stinger
string4: Hello

string1 > string4
string1.compare( string2 ) > 0
string1.compare( 2, 5, string3, 0, 5 ) == 0
string4.compare( 0, string2.length(), string2 ) == 0
string2.compare( 0, 3, string4 ) < 0
```

# 18.4 Substrings

- Member function `substr`
  - Retrieves a substring from a `string`
    - Returns a new `string` object copied from the source `string`
  - First argument
    - Specifies beginning subscript of desired substring
  - Second argument
    - Specifies length of desired substring

```cpp
// Fig. 18.3: Fig18_03.cpp
// Demonstrating string member function substr.
#include <iostream>
using std::cout;
using std::endl;

#include <string>
using std::string;

int main()
{
   string string1( "The airplane landed on time." );

   // retrieve substring "plane" which
   // begins at subscript 7 and consists of 5 elements
   cout << string1.substr( 7, 5 ) << endl;
   return 0;
} // end main
```

```
plane
```

Retrieve a substring from **`string1`**

Outline

Fig18_03.cpp

(1 of 1)

# 18.5 Swapping `strings`

- **Member function `swap`**
  - **Swaps contents of the current `string` and the argument `string`**
  - **Useful for implementing programs that sort strings**

```
1  // Fig. 18.4: Fig18_04.cpp
2  // Using the swap function to swap two strings.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <string>
8  using std::string;
9
10 int main()
11 {
12    string first( "one" );
13    string second( "two" );
14
15    // output strings
16    cout << "Before swap:\n first: " << first << "\nsecond: " << second;
17
18    first.swap( second ); // swap strings
19
20    cout << "\n\nAfter swap:\n first: " << first
21       << "\nsecond: " << second << endl;
22    return 0;
23 } // end main
```

```
Before swap:
 first: one
second: two

After swap:
 first: two
second: one
```

Outline

Fig18_04.cpp

(1 of 1)

Swap the values of **first** and **second**

# 18.7 Finding Strings and Characters in a `string`

- **Member function `find`**
  - **Attempts to find specified string in the current `string`**
    - **Returns starting location of the string if found**
    - **Returns the value `string::npos` otherwise**
      - All `string` `find`-related functions return this `const static` value to indicate the target was not found
- **Member function `rfind`**
  - **Searches current `string` backward (right-to-left) for the specified string**
    - **If the string is found, its subscript location is returned**

# 18.7 Finding Strings and Characters in a `string` (Cont.)

- Member function `find_first_of`
  - Locates first occurrence in the current `string` of any character in the specified string
- Member function `find_last_of`
  - Locates last occurrence in the current `string` of any character in the specified string
- Member function `find_first_not_of`
  - Locates first occurrence in the current `string` of any character not contained in the specified string

Outline

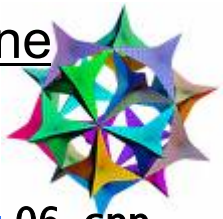

Fig18_06.cpp

(1 of 2)

```
1  // Fig. 18.6: Fig18_06.cpp
2  // Demonstrating the string find member functions.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <string>
8  using std::string;
9
10 int main()
11 {
12    string string1( "noon is 12 pm; midnight is not." );
13    int location;
14
15    // find "is" at location 5 and 25
16    cout << "Original string:\n" << string1
17       << "\n\n(find) \"is\" was found at: " << string1.find( "is" )
18       << "\n(rfind) \"is\" was found at: " << string1.rfind( "is" );
19
20    // find 'o' at location 1
21    location = string1.find_first_of( "misop" );
22    cout << "\n\n(find_first_of) found '" << string1[ location ]
23       << "' from the group \"misop\" at: " << location;
24
25    // find 'o' at location 29
26    location = string1.find_last_of( "misop" );
27    cout << "\n\n(find_last_of) found '" << string1[ location ]
28       << "' from the group \"misop\" at: " << location;
```

Attempt to find **"is"** in **string1** using function **find**

Search **string1** backward for **"is"**

Locate the first occurrence in **string1** of any character in **"misop"**

Find the last occurrence in **string1** of any character in **"misop"**

```
Original string:
noon is 12 pm; midnight is not.

(find) "is" was found at: 5
(rfind) "is" was found at: 25

(find_first_of) found 'o' from the group "misop" at: 1

(find_last_of) found 'o' from the group "misop" at: 29
```

```
29
30    // find '1' at location 8
31    location = string1.find_first_not_of( "noi spm" );
32    cout << "\n\n(find_first_not_of) '" << string1[ location ]
33       << "' is not contained in \"noi spm\" and was found at:"
34       << location;
35
36    // find '.' at location 12
37    location = string1.find_first_not_of( "12noi spm" );
38    cout << "\n\n(find_first_not_of) '" << string1[ location ]
39       << "' is not contained in \"12noi spm\" and was "
40       << "found at:" << location << endl;
41
42    // search for characters not in string1
43    location = string1.find_first_not_of(
44       "noon is 12 pm; midnight is not." );
45    cout << "\nfind_first_not_of(\"noon is 12 pm; midnight is not.\")"
46       << " returned: " << location << endl;
47    return 0;
48 } // end main
```

```
(find_first_not_of) '1' is not contained in "noi spm" and was found at:8

(find_first_not_of) '.' is not contained in "12noi spm" and was found at:12

find_first_not_of("noon is 12 pm; midnight is not.") returned: -1
```

Outline

Find the first character in **string1** not contained in the string argument

(2 of 2)

**string1** contains only characters specified in the string argument, so **string::npos** is returned