CSCI 390 – Special Topics in C++

Lecture 17 (10/16/18)

Time To Turn Off Cell Phones

C++ STL
std::string

- Header: #include <string>
- Iterator: Random Access
- Purpose: Works like strings in other language, but no garbage collection.
- Lot's of handy constructors.

std::string Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};

  // operator << overload already provided.
  cout << Hi << endl;

  // operator+ does concatenation
  cout << Hi + " World!" << endl;

  // operator+= implemented.
  Hi += " World!";
  cout << Hi << endl;

  return 0;
}
```

```
Console Log:
Hello
Hello World!
Hello World!
```

std::string/for Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hi"};

  for(auto i = 0u; i < Hi.size(); ++i)
  {
    cout << "Char: " << Hi[i] << endl;
  }

  return 0;
}
```

```
Console Log:
Char: H
Char: i
```

Range-Based for

- Syntax:
  `for ( <range_declaration> : <range_expression> ) <loop_body>`
- Handy for containers.

std::string/for Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hi"};

  for (auto c: Hi)
  {
    cout << "Char: " << c << endl;
    // c is a copy, not a reference!
    c = 'x';
  }

  // So, no changes made to Hi
  cout << "First Hi: " << Hi << endl;

  for (auto &c: Hi)
  {
    // c is a reference.  This changes Hi.
    c = 'x';
  }

  cout << "Second Hi: " << Hi << endl;

  return 0;
}
```

```
Console Log:
Char: H
Char: i
First Hi: Hi
Second Hi: xx
```

std::string/for Iterator Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};

  // Constant iterator
  for (auto it = Hi.cbegin(); it != Hi.cend(); ++it)
  {
    cout << "Char: " << *it << endl;
  }

  cout << "After constant iteration: " << Hi << endl;

  for (auto it = Hi.begin(); it != Hi.end(); ++it)
  {
    *it = 'x';
  }

  cout << "After non-constant iteration: " << Hi <<
endl;

  return 0;
}
```

```
Console Log:
Char: H
Char: e
Char: l
Char: l
Char: o
After constant iteration: Hello
After non-constant iteration:
xxxxx
```

std::string/for Reverse Iterator Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};

  // Constant reverse iterator
  for (auto it = Hi.crbegin(); it != Hi.crend(); ++it)
  {
    cout << "Char: " << *it << endl;
  }

  cout << "After constant reverse iteration: " << Hi
<< endl;

  for (auto it = Hi.rbegin(); it != Hi.rend(); ++it)
  {
    *it = 'x';
  }

  cout << "After non-constant reverse iteration: " <<
Hi << endl;

  return 0;
}
```

```
Console Log:

Char: o
Char: l
Char: l
Char: e
Char: H
After constant reverse iteration: Hello
After non-constant reverse iteration:
xxxxx
```

std::string/find Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};

  // Finding a string.
  auto FindLL = Hi.find("ll");
  if (FindLL != string::npos)
  {
    cout << "Found ll at position: " << FindLL <<
endl;
  }
  else
  {
    cout << "Did not find ll: " << FindLL << endl;
  }

  return 0;
}
```

```
Console Log:
Found ll at position: 2
```

std::string/find Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};

  // Finding a string.
  auto FindXX = Hi.find("xx");
  if (FindXX != string::npos)
  {
    cout << "Found xx at position: " << FindXX << endl;
  }
  else
  {
    cout << "Did not find xx: " << FindXX << endl;
  }

  return 0;
}
```

```
Console Log:
Did not find xx:
18446744073709551615
```

std::string/substr Example

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};

  cout << "Substr ll: " << Hi.substr(Hi.find("ll"), 2) <<
endl;
  cout << "Substr ll: " << Hi.substr(Hi.find("ll")) <<
endl;

  return 0;
}
```

```
Console Log:
Substr ll: ll
Substr ll: llo
```

std::string Member Summary

Iterators:

begin Return iterator to beginning

end Return iterator to end

rbegin Return reverse iterator to reverse beginning

rend Return reverse iterator to reverse end

cbegin Return const_iterator to beginning

cend Return const_iterator to end

crbegin Return const_reverse_iterator to reverse beginning

crend Return const_reverse_iterator to reverse end

std::string Member Summary

Capacity:

size Return length of string

length Return length of string

max_size Return maximum size of string

resize Resize string

capacity Return size of allocated storage

reserve Request a change in capacity

clear Clear string

empty Test if string is empty

shrink_to_fit Shrink to fit

std::string Member Summary

Element access:

operator[ ] Get character of string

at Get character in string

back Access last character

front Access first character

std::string Member Summary

Modifiers:

operator+= Append to string

append Append to string

push_back Append character to string

assign Assign content to string

insert Insert into string

erase Erase characters from string

replace Replace portion of string

swap Swap string values

pop_back Delete last character

std::string Member Summary

String operations:

c_str Get C string equivalent

data Get string data

copy Copy sequence of characters from string

find Find content in string

rfind Find last occurrence of content in string

find_first_of Find character in string

find_last_of Find character in string from the end

find_first_not_of Find absence of character in string

find_last_not_of Find non-matching character in string from the end

substr Generate substring

compare Compare strings

std::string Member Summary

Member constants:

npos Maximum value for size_t

std::string Member Summary

Non-member function overloads:

operator+ Concatenate strings (function)

swap Exchanges the values of two strings (function)

operator>> Extract string from stream (function)

operator<< Insert string into stream (function)

getline Get line from stream into string (function)

Hello <algorithm>

- #include <algorithm> gives you access to a library of handy algorithms:
  - Many use iterators.
  - Some use lambda functions.

Algorithms: copy/copy_if

- Headers:

```
template< class InputIt, class OutputIt >
OutputIt copy( InputIt first, InputIt last, OutputIt d_first );

template< class InputIt, class OutputIt, class UnaryPredicate >
OutputIt copy_if( InputIt first, InputIt last,
                  OutputIt d_first,
                  UnaryPredicate pred );
```

Algorithms: copy

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

#include <algorithm>
using std::copy;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};
  string Hi2{"There"};

  cout << " Before copy: Hi2: " << Hi2 << endl;

  copy(Hi.cbegin(), Hi.cend(), Hi2.begin());

  cout << "  After copy: Hi2: " << Hi2 << endl;

  copy(Hi.cbegin(), Hi.cend(), Hi2.rbegin());

  cout << " After rcopy: Hi2: " << Hi2 << endl;

  return 0;
}
```

```
Console Log:
 Before copy: Hi2: There
  After copy: Hi2: Hello
 After rcopy: Hi2: olleH
```

Algorithms: copy_if

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

#include <algorithm>
using std::copy; using std::copy_if;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};
  string Hi2{"There"};

  cout << "  Before copy: Hi2: " << Hi2 << endl;

  copy_if(Hi.cbegin(), Hi.cend(), Hi2.begin(), [](const
char &c){return c != 'e';});

  cout << "After copy_if: Hi2: " << Hi2 << endl;

  copy(Hi.cbegin(), Hi.cend(), Hi2.rbegin());

  cout << "  After rcopy: Hi2: " << Hi2 << endl;

  return 0;
}
```

```
Console Log:
  Before copy: Hi2: There
After copy_if: Hi2: Hlloe
  After rcopy: Hi2: olleH
```

Algorithms: copy_if: Cleaning UP

```
#include <iostream>
using std::cout;  using std::endl;

#include <string>
using std::string;

#include <algorithm>
using std::copy; using std::copy_if;

int main(void)
{
  // Common constructor
  string Hi{"Hello"};
  string Hi2{"There"};

  cout << "  Before copy: Hi2: " << Hi2 << endl;

  string::iterator it = copy_if(Hi.cbegin(), Hi.cend(),
Hi2.begin(), [](const char &c){return c != 'e';});
  Hi2.erase(it, Hi2.end());

  cout << "After copy_if: Hi2: " << Hi2 << endl;

  copy(Hi.cbegin(), Hi.cend(), Hi2.rbegin());

  cout << "  After rcopy: Hi2: " << Hi2 << endl;

  return 0;
}
```

```
Console Log:
  Before copy: Hi2: There
  After copy_if: Hi2: Hllo
  After rcopy: Hi2: lleH
```