CSCI 390 – Special Topics in C++

Lecture 8

9/13/18

Time To Turn Off Cell Phones

Operators (cont)
Bitwise Operators

```
#include <iostream>
#include <bitset>

typedef std::bitset<8> tBits;

int main()
{
  tBits lhs{"00111100"};
  tBits rhs{"01011010"};

  std::cout << "   " << lhs << std::endl;
  std::cout << " ~ " << ~lhs << "\n" << std::endl;

  std::cout << "   " << rhs << std::endl;
  std::cout << " ~ " << ~rhs << "\n" << std::endl;

  std::cout << "   " << rhs << std::endl;
  std::cout << " & " << lhs <<  std::endl;
  std::cout << "   " << "--------" << std::endl;
  std::cout << "   " << (lhs & rhs) << "\n" << std::endl;

  std::cout << "   " << rhs << std::endl;
  std::cout << " | " << lhs <<  std::endl;
  std::cout << "   " << "--------" << std::endl;
  std::cout << "   " << (lhs | rhs) << "\n" << std::endl;

  std::cout << "   " << rhs << std::endl;
  std::cout << " ^ " << lhs <<  std::endl;
  std::cout << "   " << "--------" << std::endl;
  std::cout << "   " << (lhs ^ rhs) << "\n" << std::endl;

  return 0;
}
```

```
   00111100
 ~ 11000011

   01011010
 ~ 10100101

   01011010
 & 00111100
   --------
   00011000

   01011010
 | 00111100
   --------
   01111110

   01011010
 ^ 00111100
   --------
   01100110


...Program finished with exit code 0
Press ENTER to exit console.
```

Operators (cont)
Priority 3, Right To Left

- **!**<rvalue> (Logical not)
  - Returns: **bool**, 1 if <rvalue> == 0, 0 otherwise
  - Side Effects: None
- **~**<rvalue> (Bitwise not)
  - Returns: <rvalue> with bits flipped
  - Side Effects: None

Operators (cont)
Priority 3, Right To Left

- **+**<rvalue> (Unary +)
  - Returns: <rvalue>
  - Side Effects: None
- **–**<rvalue> (Unary -)
  - Returns: -<rvalue>
  - Side Effects: None

Operators (cont)
Priority 2, Left to Right

- **++**<lvalue> (Pre-increment)
  - Returns: <lvalue> + 1 (This is an <rvalue>.)
  - Side Effects: Increments <lvalue>
- **--**<lvalue> (Pre-decrement)
  - Returns: <lvalue> - 1 (This is an <rvalue>.)
  - Side Effects: Decrements <lvalue>

Operators (cont)
Priority 3, Right To Left

- ∗<rvalue> (Dereference)
  - Returns: Returns object pointed to by <rvalue> (This is a <lvalue>.
  - Side Effects: None
- &<lvalue> (Address of)
  - Returns: Address where <lvalue> is stored. (This is a <rvalue>
  - Side Effects: None

Operators (cont)
Priority 3, Right To Left

- **`sizeof`**`(`<rvalue> | <type>`)`
  - Returns: Returns the number of bytes required to store the <rvalue> or <type>. (This is a **`size_t`** <rvalue>.
  - Side Effects: None
- <type>`(`<rvalue>`)` (Cast)
  - Returns: Returns <rvalue> with type changed to <type>.
  - Side Effects: None

Operators (cont)
Priority 5, Left To Right

- <lhs rvalue> * <rhs rvalue>
  - Returns: Returns <lhs rvalue> * <rhs rvalue>.
  - Side Effects: None
- <lhs rvalue> / <rhs rvalue>
  - Returns: Returns <lhs rvalue> / <rhs rvalue>.
  - Side Effects: None
- <lhs rvalue> % <rhs rvalue>
  - Returns: Returns <lhs rvalue> modulo <rhs rvalue>.
  - Side Effects: None

Operators (cont)
Priority 6, Left To Right

- <lhs rvalue> **+** <rhs rvalue>
  - Returns: Returns <lhs rvalue> + <rhs rvalue>.
  - Side Effects: None
- <lhs rvalue> **–** <rhs rvalue>
  - Returns: Returns <lhs rvalue> - <rhs rvalue>.
  - Side Effects: None

Operators (cont)
Priority 7, Left To Right

- <lhs rvalue> **<<** <rhs rvalue>
  - Returns: Returns <lhs rvalue> shifted left <rhs rvalue> bits. Bits shifted out the high order end are lost. Zeros are shifted into the low order end.
  - Side Effects: None
- <lhs rvalue> **>>** <rhs rvalue>
  - Returns: Returns <lhs rvalue> shifted right <rhs rvalue> bits. Bits shifted out the low order end are lost. Zeros are shifted into the high order end.
  - Side Effects: None

Operators (cont)
<< and >> Examples

```
#include <iostream>
#include <bitset>

typedef std::bitset<8> tBits;

int main()
{
  tBits lhs{"01011010"};

  std::cout << "01011010 << 3 = " <<  (lhs << 3) << std::endl;
  std::cout << "01011010 >> 3 = " <<  (lhs >> 3) << std::endl;

  return 0;
}
```

```
01011010 << 3 = 11010000
01011010 >> 3 = 00001011


...Program finished with exit code 0
Press ENTER to exit console.
```

Operators (cont)
Priority 8, Left To Right

- <lhs rvalue> **<** <rhs rvalue>
  - Returns: Returns **bool**.
    true when <lhs rvalue> **<** <rhs rvalue>, else **false**.
  - Side Effects: None
- <lhs rvalue> **<=** <rhs rvalue>
  - Returns: Returns **bool**.
    true when <lhs rvalue> **<=** <rhs rvalue>, else **false**.
  - Side Effects: None

Operators (cont)
Priority 8, Left To Right

- <lhs rvalue> **>** <rhs rvalue>
  - Returns: Returns **bool**.
    true when <lhs rvalue> **>** <rhs rvalue>, else **false**.
  - Side Effects: None
- <lhs rvalue> **>=** <rhs rvalue>
  - Returns: Returns **bool**.
    true when <lhs rvalue> **>=** <rhs rvalue>, else **false**.
  - Side Effects: None

Operators (cont)
Priority 9, Left To Right

- <lhs rvalue> **==** <rhs rvalue>
  - Returns: Returns **bool**.
    true when <lhs rvalue> **==** <rhs rvalue>, else **false**.
  - Side Effects: None
- <lhs rvalue> **!=** <rhs rvalue>
  - Returns: Returns **bool**.
    true when <lhs rvalue> **!=** <rhs rvalue>, else **false**.
  - Side Effects: None

Operators (cont)
Priority 10, Left To Right

- <lhs rvalue> **&** <rhs rvalue>
  - Returns: Returns bitwise boolean AND of its operands. Each bit is computed according to this truth table:

p	q	p&q
T	T	T
T	F	F
F	T	F
F	F	F

  - Side Effects: None

Operators (cont)
Priority 11, Left To Right

- <lhs rvalue> ^ <rhs rvalue>
  - Returns: Returns bitwise boolean XOR of its operands. Each bit is computed according to this truth table:

p	q	p^q
T	T	F
T	F	T
F	T	T
F	F	F

  - Side Effects: None

Operators (cont)
Priority 12, Left To Right

- <lhs rvalue> | <rhs rvalue>
  - Returns: Returns bitwise boolean OR of its operands. Each bit is computed according to this truth table:

p	q	p | q
T	T	T
T	F	T
F	T	T
F	F	F

  - Side Effects: None

Operators (cont)
Priority 13, Left To Right

- <lhs rvalue> **&&** <rhs rvalue>
  - Returns: **`bool`** compute as follows:
    - Evaluate <lhs rvalue>. If 0 return false, else evaluate and return **`false`** if <rhs rvalue> == 0 else **`true`**.
    - Note <rhs rvalue> is NOT evaluated if <lhs rvalue> is 0.
    - This is called short-circuit evaluation, minimal evaluation, or McCarthy evaluation.
      - John McCarthy introduced this in 1958 with the first version of LISP.
    - Think of it as AND IF.
  - Side Effects: None

Operators (cont)
&& Examples

```
#include <iostream>

int main()
{
  auto x = 1.5;

  bool rvalue1 = 0 && (x = 0.0);

  std::cout << "rvalue1: " << rvalue1 << std::endl;
  std::cout << "x: " << x << std::endl;

  bool rvalue2 = 1 && (x = 1.0);

  std::cout << "rvalue2: " << rvalue2 << std::endl;
  std::cout << "x: " << x << std::endl;

  bool rvalue3 = 1 && (x = 0.0);

  std::cout << "rvalue3: " << rvalue3 << std::endl;
  std::cout << "x: " << x << std::endl;


  return 0;
}
```

```
rvalue1: 0
x: 1.5
rvalue2: 1
x: 1
rvalue3: 0
x: 0


...Program finished with exit code 0
Press ENTER to exit console.
```

Operators (cont)
Priority 14, Left To Right

- <lhs rvalue> || <rhs rvalue>
  - Returns: **`bool`** compute as follows:
    - Evaluate <lhs rvalue>. If 1 return true, else evaluate and return **`true`** if <rhs rvalue> != 0 else **`false`**.
    - Note <rhs rvalue> is NOT evaluated if <lhs rvalue> is 1.
    - This is called short-circuit evaluation, minimal evaluation, or McCarthy evaluation.
      - John McCarthy introduced this in 1958 with the first version of LISP.
    - Think of it as OR IF.
  - Side Effects: None

Operators (cont)
| | Examples

```
#include <iostream>

int main()
{
  auto x = 1.5;

  bool rvalue1 = 1 || (x = 0.0);

  std::cout << "rvalue1: " << rvalue1 << std::endl;
  std::cout << "x: " << x << std::endl;

  bool rvalue2 = 0 || (x = 1.0);

  std::cout << "rvalue2: " << rvalue2 << std::endl;
  std::cout << "x: " << x << std::endl;

  bool rvalue3 = 0 || (x = 0.0);

  std::cout << "rvalue3: " << rvalue3 << std::endl;
  std::cout << "x: " << x << std::endl;


  return 0;
}
```

```
rvalue1: 1
x: 1.5
rvalue2: 1
x: 1
rvalue3: 0
x: 0


...Program finished with exit code 0
Press ENTER to exit console.
```

Operators (cont)
Priority 15, Right To Left

- <rvalue1> **?** <rvalue2> **:** <rvalue3>
  - Returns: Evaluate <rvalue1>. If 1, then return <rvalue2>, else return <rvalue3>.
  - Note: <rvalue2> and <rvalue3> must be the same type.
  - Side Effects: None

Operators (cont)
?: Examples

```
#include <iostream>

int main()
{
  auto x = 1.5;

  std::cout << "x: " << x << std::endl;
  std::cout << "1.0 is" << (1.0 < x ? "" : " not") <<
    " < x" << std::endl;
  std::cout << "2.0 is" << (2.0 < x ? "" : " not") <<
    " < x" << std::endl;

  return 0;
}
```

```
x: 1.5
1.0 is < x
2.0 is not < x


...Program finished with exit code 0
Press ENTER to exit console.
```

Operators (cont)
Priority 16, Right To Left

- <lvalue> **=** <rvalue>
  - Returns: The value stored at <lvalue>, i.e., <rvalue>.
  - Side Effects: Stores <rvalue> at <lvalue>

Operators (cont)
= Example

```
#include <iostream>

int main()
{
  auto x = 1.5;
  auto y = 2.5;

  std::cout << "x: " << x << std::endl;
  std::cout << "y: " << y << std::endl;

  x = y = 3.5;

  std::cout << "x: " << x << std::endl;
  std::cout << "y: " << y << std::endl;

  return 0;
}
```

```
x: 1.5
y: 2.5
x: 3.5
y: 3.5


...Program finished with exit code 0
Press ENTER to exit console.
```

Operators (cont)
Priority 16, Right To Left

- <lvalue> **+=|-=|*=|/=|%=|>>=|<<=|&=|^=||=** <rvalue>
  - Returns: <lvalue> = <rhs>, where:
  - <rhs> is <lvalue> **+|-|*|/|%|>>|<<|&|^||**`<rvalue>`.
  - Side Effects: Stores <rhs> at <lvalue>

Operators (cont)
= Example

```
#include <iostream>

int main()
{
  auto x = 1.5;
  auto y = 2.5;

  std::cout << "x: " << x << std::endl;
  std::cout << "y: " << y << std::endl;

  x = y += 1.0;

  std::cout << "x: " << x << std::endl;
  std::cout << "y: " << y << std::endl;

  return 0;
}
```

```
x: 1.5
y: 2.5
x: 3.5
y: 3.5


...Program finished with exit code 0
Press ENTER to exit console.
```

Operators (cont)
Priority 17, Left To Right

- <exp1>, <exp2>, . . . , <exp*n*>
  - Returns: <exp*n*>
  - Note: Each expresssion is evaluated left to right.
  - Side Effects: None.

Operators (cont)
, Examples

```
#include <iostream>

int main()
{
  auto x = 1.5;
  auto y = 2.5;

  std::cout << "x: " << x << std::endl;
  std::cout << "y: " << y << std::endl;

  auto z = (x -= 0.5, 11, x + y + 1.0, -1.5);

  std::cout << "x: " << x << std::endl;
  std::cout << "y: " << y << std::endl;
  std::cout << "z: " << z << std::endl;

  y = (x -= 0.5, x + y + 1.0);

  std::cout << "x: " << x << std::endl;
  std::cout << "y: " << y << std::endl;
  std::cout << "z: " << z << std::endl;

  return 0;
}
```

```
x: 1.5
y: 2.5
x: 1
y: 2.5
z: -1.5
x: 0.5
y: 4
z: -1.5


...Program finished with exit code 0
Press ENTER to exit console.
```