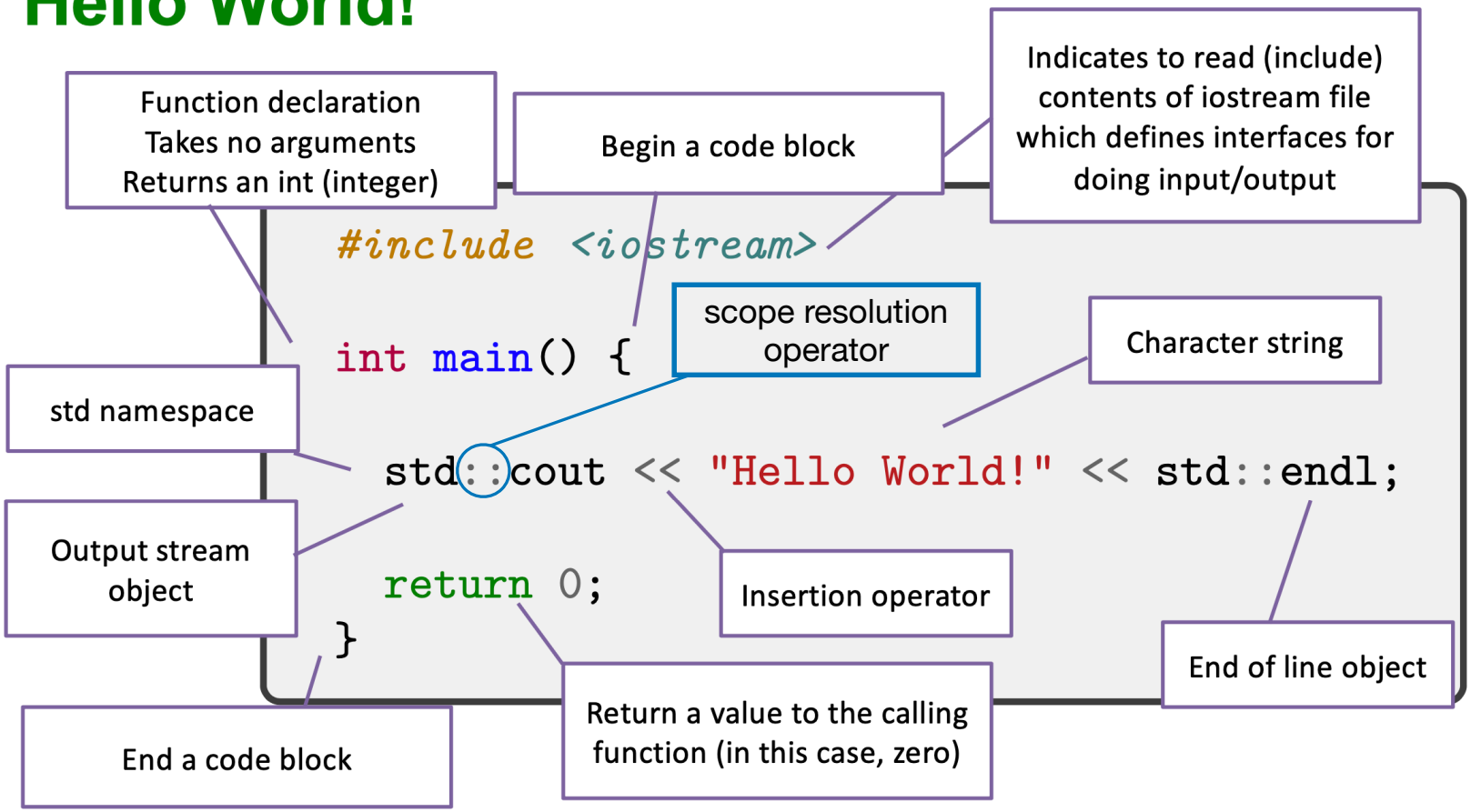
lecture 4

- C++ at a Glance
  - Visual Studio Code -demo
  - simple but useful starter code
  - basic data types (and their sizes)
  - scope, scope resolution operator
  - namespaces
- Compiling 1

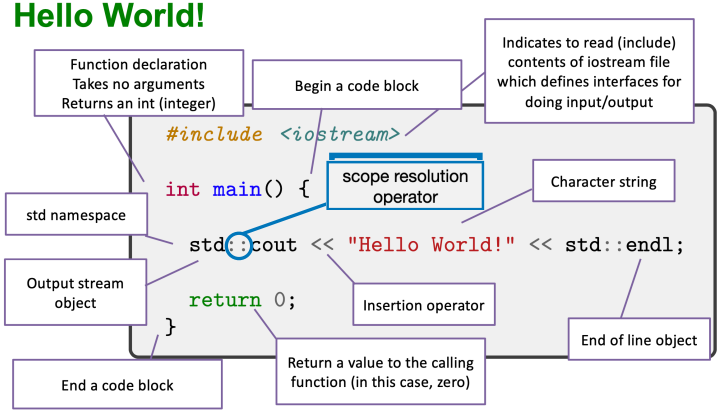
Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

Hello World!



Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

Hello World!



- Visual Studio Code -demo
- compile errors
  - undeclared identifiers
- compile warnings
  - implicit conversion (i.e., return float instead of int)

```
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% ./xhello
Hello, World!
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% echo $?
0
~/Desktop/kr-uw/amath83-spr2025/lecture-codes%
```

Variables

- data types
  - cout << sizeof(<type>)
  - int
    - int8_t, int16_t, int32_t, int64_t
    - short, long, unsigned
  - float
    - double, long double
- declaring / assigning vars
- scope
  - local, global
- signed / unsigned numbers
- literals (50: octal 062, hex 0x32)

Data Type	Size	Description
`boolean`	1 byte	Stores true or false values
`char`	1 byte	Stores a single character/letter/number, or ASCII values
`int`	2 or 4 bytes	Stores whole numbers, without decimals
`float`	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits
`double`	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits

Syntax

```
type variableName = value;
```

Kenneth J. Roche
University of Washington

C++ type sizes

```
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% ls -lstr
total 16
8 -rw-r--r--@ 1 kennethroche  staff   238 Apr  6 22:09 hello-world.cpp
8 -rw-r--r--@ 1 kennethroche  staff  1080 Apr  6 22:31 cpp-types.cpp
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% g++ -std=c++17 -c cpp-types.cpp
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% ls -lstr
total 48
 8 -rw-r--r--@ 1 kennethroche  staff    238 Apr  6 22:09 hello-world.cpp
 8 -rw-r--r--@ 1 kennethroche  staff   1080 Apr  6 22:31 cpp-types.cpp
32 -rw-r--r--  1 kennethroche  staff  13168 Apr  6 22:35 cpp-types.o
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% g++ -o xcpp-types cpp-types.o
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% ls -lstr
total 128
 8 -rw-r--r--@ 1 kennethroche  staff    238 Apr  6 22:09 hello-world.cpp
 8 -rw-r--r--@ 1 kennethroche  staff   1080 Apr  6 22:31 cpp-types.cpp
32 -rw-r--r--  1 kennethroche  staff  13168 Apr  6 22:35 cpp-types.o
80 -rwxr-xr-x  1 kennethroche  staff  39512 Apr  6 22:35 xcpp-types
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% ./xcpp-types
Sizes of various data types (in bytes):
int8_t:        1
int16_t:       2
int32_t:       4
int64_t:       8
unsigned int:  4
short:         2
int:           4
long:          8
long long:     8
float:         4
double:        8
long double:   8
char:          1
bool:          1
~/Desktop/kr-uw/amath83-spr2025/lecture-codes%
```

```cpp
#include <iostream>
#include <cstdint> // for int8_t, etc.

int main()
{
    std::cout << "Sizes of various data types (in bytes):\n";

    // Integers
    std::cout << "int8_t:        " << sizeof(int8_t) << '\n';
    std::cout << "int16_t:       " << sizeof(int16_t) << '\n';
    std::cout << "int32_t:       " << sizeof(int32_t) << '\n';
    std::cout << "int64_t:       " << sizeof(int64_t) << '\n';
    std::cout << "unsigned int:  " << sizeof(unsigned int) << '\n';
    std::cout << "short:         " << sizeof(short) << '\n';
    std::cout << "int:           " << sizeof(int) << '\n';
    std::cout << "long:          " << sizeof(long) << '\n';
    std::cout << "long long:     " << sizeof(long long) << '\n';

    // Floating-point
    std::cout << "float:         " << sizeof(float) << '\n';
    std::cout << "double:        " << sizeof(double) << '\n';
    std::cout << "long double:   " << sizeof(long double) << '\n';

    // Other
    std::cout << "char:          " << sizeof(char) << '\n';
    std::cout << "bool:          " << sizeof(bool) << '\n';

    return 0;
}
```

C++ at a glance

Operators

- int
  - +, -, *, /, % (modulus division remainder)
- float
  - +, -, *, /
  - 2.0f
  - static_cast<float>(2)
- increment, decrement
  - i++; // i = i+1; post: returns i before change
  - i--; // i = i-1;
  - --i; ++i; // pre: changes i before return

- boolean
  - bool b = (2<3); // b is false
  - ==,!=,>,<,>=,<=
  - || // logical or
  - && // logical and
- bitwise
  - & (and), | (or), ^(xor)
  - <<(l shift), >>(rshift)
  - ~ (not, invert)

Kenneth J. Roche
University of Washington

```cpp
#include <iostream>

int main()
{
    std::cout << "=== Integer Operators ===\n";
    int a = 10, b = 3;
    std::cout << "a + b = " << a + b << '\n';
    std::cout << "a - b = " << a - b << '\n';
    std::cout << "a * b = " << a * b << '\n';
    std::cout << "a / b = " << a / b << " (integer division)\n";
    std::cout << "a % b = " << a % b << " (modulus)\n";

    std::cout << "\n=== Float Operators ===\n";
    float x = 10.0f, y = 3.0f;
    std::cout << "x + y = " << x + y << '\n';
    std::cout << "x - y = " << x - y << '\n';
    std::cout << "x * y = " << x * y << '\n';
    std::cout << "x / y = " << x / y << '\n';
    float z = static_cast<float>(2);
    std::cout << "static_cast<float>(2) = " << z << '\n';

    std::cout << "\n=== Increment / Decrement ===\n";
    int i = 5;
    std::cout << "i = " << i << '\n';
    std::cout << "i++ = " << i++ << " (post-increment)\n";
    std::cout << "After i++: i = " << i << '\n';
    std::cout << "++i = " << ++i << " (pre-increment)\n";
    std::cout << "i-- = " << i-- << " (post-decrement)\n";
    std::cout << "After i--: i = " << i << '\n';
    std::cout << "--i = " << --i << " (pre-decrement)\n";

    std::cout << "\n=== Boolean and Comparison ===\n";
    bool b1 = (2 < 3);
    std::cout << "2 < 3 = " << b1 << '\n';
    std::cout << "2 == 2: " << (2 == 2) << '\n';
    std::cout << "2 != 3: " << (2 != 3) << '\n';
    std::cout << "5 > 3: " << (5 > 3) << '\n';
    std::cout << "5 <= 5: " << (5 <= 5) << '\n';
    std::cout << "(true || false): " << (true || false) << '\n';
    std::cout << "(true && false): " << (true && false) << '\n';

    std::cout << "\n=== Bitwise Operators ===\n";
    int p = 6;                                                    // 0110
    int q = 3;                                                    // 0011
    std::cout << "p & q = " << (p & q) << " (bitwise AND)\n";     // 0010
    std::cout << "p | q = " << (p | q) << " (bitwise OR)\n";      // 0111
    std::cout << "p ^ q = " << (p ^ q) << " (bitwise XOR)\n";     // 0101
    std::cout << "p << 1 = " << (p << 1) << " (left shift)\n";   // 1100 = 12
    std::cout << "p >> 1 = " << (p >> 1) << " (right shift)\n"; // 0011 = 3
    std::cout << "~p = " << (~p) << " (bitwise NOT)\n";

    return 0;
}
```

```cpp
#include <iostream>

int main()
{
    std::cout << "=== Integer Operators ===\n";
    int a = 10, b = 3;
    std::cout << "a + b = " << a + b << '\n';
    std::cout << "a - b = " << a - b << '\n';
    std::cout << "a * b = " << a * b << '\n';
    std::cout << "a / b = " << a / b << " (integer division)\n";
    std::cout << "a % b = " << a % b << " (modulus)\n";

    std::cout << "\n=== Float Operators ===\n";
    float x = 10.0f, y = 3.0f;
    std::cout << "x + y = " << x + y << '\n';
    std::cout << "x - y = " << x - y << '\n';
    std::cout << "x * y = " << x * y << '\n';
    std::cout << "x / y = " << x / y << '\n';
    float z = static_cast<float>(2);
    std::cout << "static_cast<float>(2) = " << z << '\n';

    std::cout << "\n=== Increment / Decrement ===\n";
    int i = 5;
    std::cout << "i = " << i << '\n';
    std::cout << "i++ = " << i++ << " (post-increment)\n";
    std::cout << "After i++: i = " << i << '\n';
    std::cout << "++i = " << ++i << " (pre-increment)\n";
    std::cout << "i-- = " << i-- << " (post-decrement)\n";
    std::cout << "After i--: i = " << i << '\n';
    std::cout << "--i = " << --i << " (pre-decrement)\n";

    std::cout << "\n=== Boolean and Comparison ===\n";
    bool b1 = (2 < 3);
    std::cout << "2 < 3 = " << b1 << '\n';
    std::cout << "2 == 2: " << (2 == 2) << '\n';
    std::cout << "2 != 3: " << (2 != 3) << '\n';
    std::cout << "5 > 3: " << (5 > 3) << '\n';
    std::cout << "5 <= 5: " << (5 <= 5) << '\n';
    std::cout << "(true || false): " << (true || false) << '\n';
    std::cout << "(true && false): " << (true && false) << '\n';

    std::cout << "\n=== Bitwise Operators ===\n";
    int p = 6;                                          // 0110
    int q = 3;                                          // 0011
    std::cout << "p & q = " << (p & q) << " (bitwise AND)\n";   // 0010
    std::cout << "p | q = " << (p | q) << " (bitwise OR)\n";    // 0111
    std::cout << "p ^ q = " << (p ^ q) << " (bitwise XOR)\n";   // 0101
    std::cout << "p << 1 = " << (p << 1) << " (left shift)\n";  // 1100 = 12
    std::cout << "p >> 1 = " << (p >> 1) << " (right shift)\n"; // 0011 = 3
    std::cout << "~p = " << (~p) << " (bitwise NOT)\n";

    return 0;
}
```

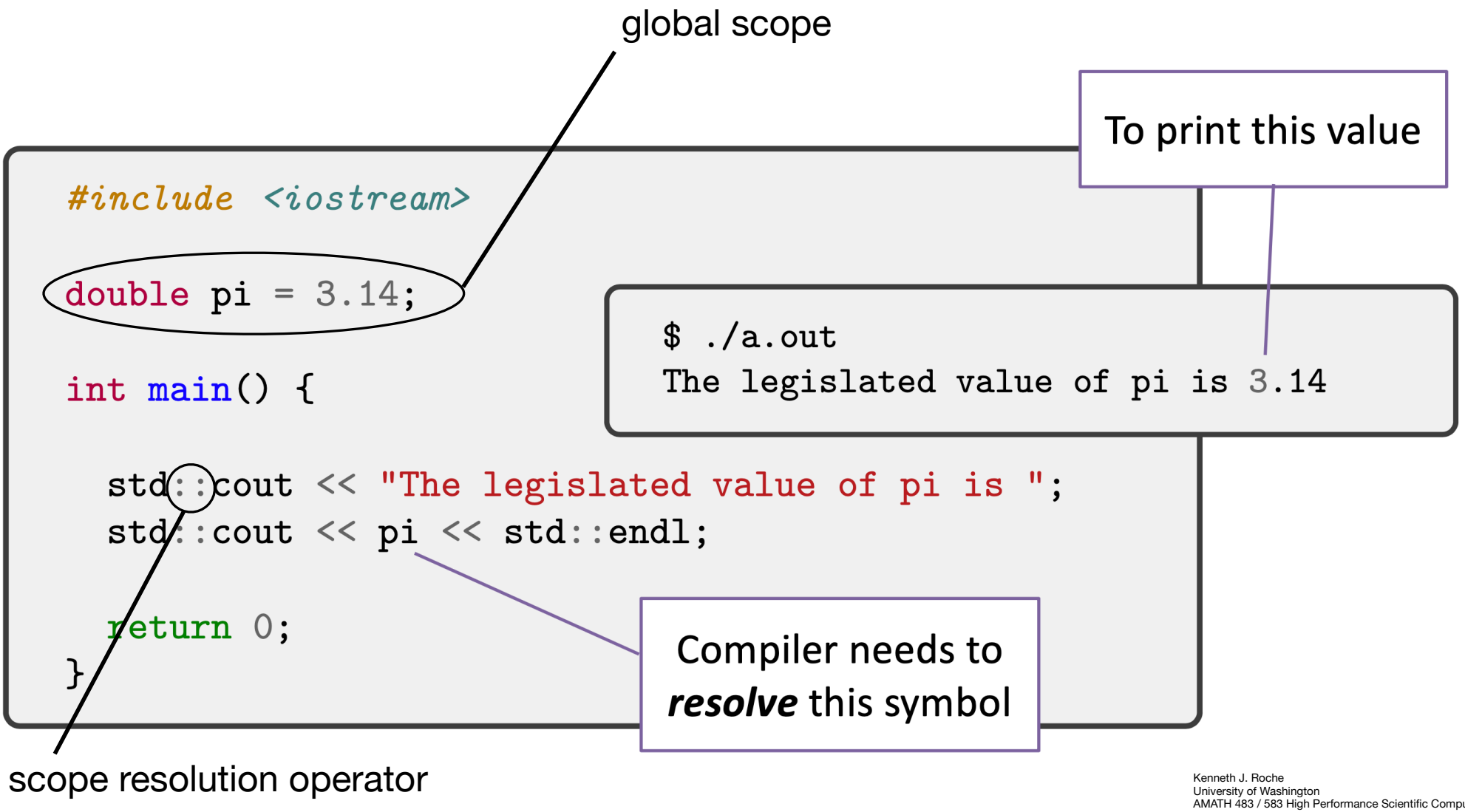
```
~/Desktop/kr-uw/amath83-spr2025/lecture-codes% ./xbasic-operations
=== Integer Operators ===
a + b = 13
a - b = 7
a * b = 30
a / b = 3 (integer division)
a % b = 1 (modulus)

=== Float Operators ===
x + y = 13
x - y = 7
x * y = 30
x / y = 3.33333
static_cast<float>(2) = 2

=== Increment / Decrement ===
i = 5
i++ = 5 (post-increment)
After i++: i = 6
++i = 7 (pre-increment)
i-- = 7 (post-decrement)
After i--: i = 6
--i = 5 (pre-decrement)

=== Boolean and Comparison ===
2 < 3 = 1
2 == 2: 1
2 != 3: 1
5 > 3: 1
5 <= 5: 1
(true || false): 1
(true && false): 0

=== Bitwise Operators ===
p & q = 2 (bitwise AND)
p | q = 7 (bitwise OR)
p ^ q = 5 (bitwise XOR)
p << 1 = 12 (left shift)
p >> 1 = 3 (right shift)
~p = -7 (bitwise NOT)
~/Desktop/kr-uw/amath83-spr2025/lecture-codes%
```

```cpp
std::cout << "\n=== Bitwise Operators ===\n";
int p = 6;                                          // 0110
int q = 3;                                          // 0011
std::cout << "p & q = " << (p & q) << " (bitwise AND)\n";   // 0010
std::cout << "p | q = " << (p | q) << " (bitwise OR)\n";    // 0111
std::cout << "p ^ q = " << (p ^ q) << " (bitwise XOR)\n";   // 0101
std::cout << "p << 1 = " << (p << 1) << " (left shift)\n";  // 1100 = 12
std::cout << "p >> 1 = " << (p >> 1) << " (right shift)\n"; // 0011 = 3
std::cout << "~p = " << (~p) << " (bitwise NOT)\n";
```

In two's complement:

- The **leftmost bit** (most significant bit, MSB) is the **sign bit**: `0` means positive, `1` means negative.
- To find the decimal value of a **negative number**, we:
  1. Invert all the bits (again)
  2. Add 1
  3. Convert the result to decimal and then **add a minus sign**

enneth J. Roche
niversity of Washington
MATH 483 / 583 High Performance Scientific Computing
pring Quarter 2025



Kenneth J. Roche
University of Washington

Namespaces

```
#include <iostream>

namespace amath583 {
  double pi = 3.14;
};

int main() {

  std::cout << "The legislated value of pi is "
  std::cout << pi << std::endl;

  return 0;
}
```

```
ns.cpp:11:16: error: use of undeclared identifier 'pi'; did you mean 'amath583::pi'?
  std::cout << pi << std::endl;
               ^~
               amath583::pi
ns.cpp:5:10: note: 'amath583::pi' declared here
  double pi = 3.14;
         ^
1 error generated.
```

pi is hidden in a namespace

But can't

Compiler needs to resolve this symbol

Namespaces

```
#include <iostream>

namespace amath583 {
  double pi = 3.14;
};

int main() {

  std::cout << "The legislated value of pi is ";
  std::cout << amath583::pi << std::endl;

  return 0;
}
```

```
$ ./a.out
The legislated value of pi is 3.14
```

Look for the variable pi in the amath583 namespace

Namespaces

```
#include <iostream>

namespace amath583 {
  double pi = 3.14;
};

using amath583::pi;

int main() {

  std::cout << "The legislated value of pi is ";
  std::cout << pi << std::endl;

  return 0;
}
```

We can also tell the compiler to look for pi in the namespace this way

```
$ ./a.out
The legislated value of pi is 3.14
```

Use “pi” where it can be found (no specified namespace)

Namespaces

```
#include <iostream>

namespace amath583 {
  double pi = 3.14;
};

using amath583;

int main() {

  std::cout << "The legislated value of pi is ";
  std::cout << pi << std::endl;

  return 0;
}
```

Everything in this namespace can be found

Open the namespace

Use “pi” where it can be found

Namespaces

```cpp
#include <iostream>

namespace amath483 {
  double pi = 3.14159;
};

namespace amath583 {
  double pi = 3.14;
};

using namespace amath483::pi;

int main() {

  std::cout << "The legislated value of pi is ";
  std::cout << pi << std::endl;

  return 0;
}
```

```
$ ./a.out
The legislated value of pi is 3.14159
```

Specify that this pi is in the global namespace

The global namespace

```
#include <iostream>
#include <iomanip>

namespace amath483 {
  double pi = 3.14159;
};

namespace amath583 {
  double pi = 3.14;
};

double pi = 3.14159265358979323846264338327950288419 7;

using namespace amath483;
using namespace amath583;

int main() {

  std::cout << "The value of pi is ";
  std::cout << std::setprecision(15) << ::pi << std::endl;

  return 0;
}
```

```
$ ./a.out
The value of pi is 3.14159265358979
```

Include iomanip

Set precision to 15 digits

```cpp
#include <iostream>

namespace amath483 {
  double pi = 3.14159;
};

namespace amath583 {
  double pi = 3.14;
};

void foo() {
  using namespace amath583;
  std::cout << "The legislated value of pi is ";
  std::cout << pi << std::endl;
}

void bar() {
  using namespace amath483;
  std::cout << "The legislated value of pi is ";
  std::cout << pi << std::endl;
}

int main() {
  foo() ; bar() ;
  return 0;
}
```

Local scope

Local scope

```cpp
#include <iostream>

using namespace std;

int main() {

  string message = "Hello World";
  cout << message << endl;

  return 0;
}
```

Open all of namespace std

```cpp
#include <iostream>
#include <string>

int main() {

  std::string message = "Hello World";
  std::cout << message << std::endl;

  return 0;
}
```

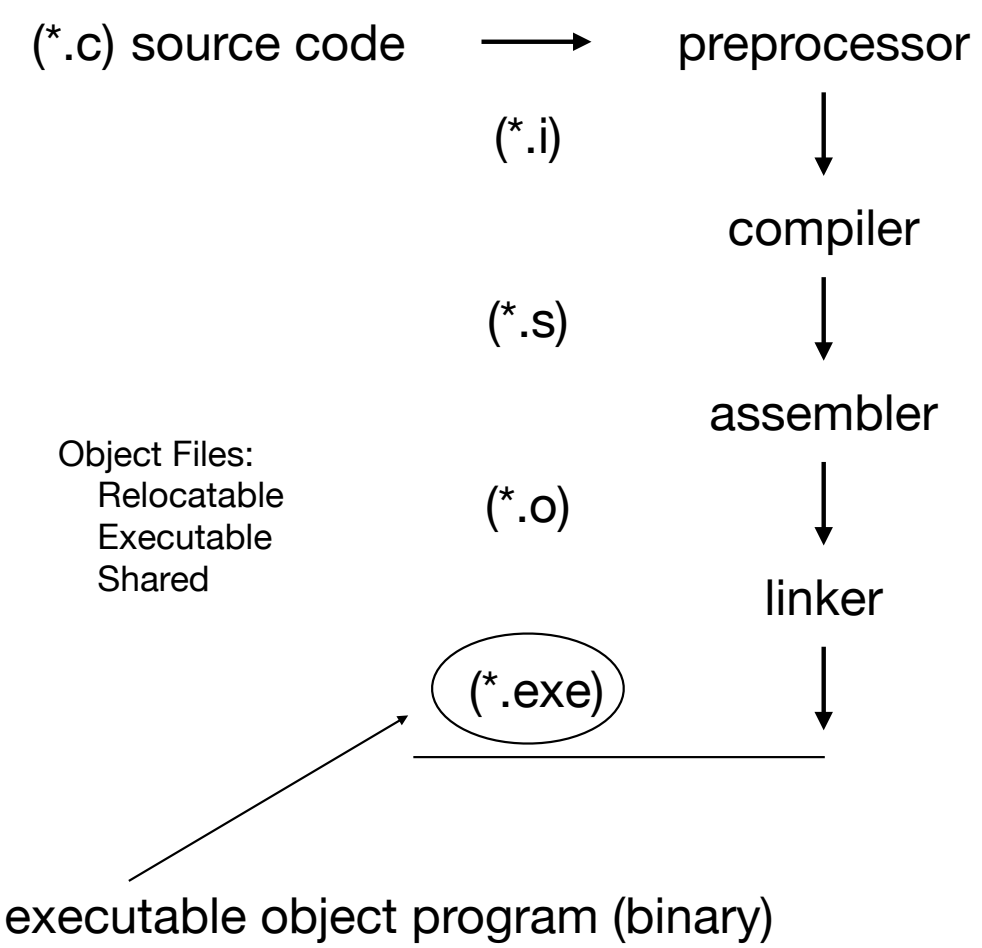
Explicitly use variables from namespace std

★ Express Intent

```cpp
#include <iostream>

using std::cout;  using std::endl;
using std::string;

int main() {

  string message = "Hello World";
  cout << message << endl;

  return 0;
}
```

Using for just the variables you want

Compiling



statements, insert include code, expand macros, etc.

translates *.i* file into *.s* file -assembly language program; machine instructions in text

translates *.s* file into *.o* file -relocatable object program; encodes assembly machine instructions into binary

combines *.o* relocatable objects with necessary system relocatable objects to create executable object program which can be loaded into memory and executed by the system

End Lecture 4

enneth J. Roche
niversity of Washington
MATH 483 / 583 High Performance Scientific Computing
pring Quarter 2025