More C++

For Loops
Scopes
Input
Randomness
Programming!!
For Loops

Python

```
for x in range(10):
    print x
```

MATLAB

```
for x = 0:9
    x
end
```

Thing we are looping over

Variable holding current value

Statements to run every time
For Loops

Python

```python
for x in range(10):
    print x
```

MATLAB

```matlab
for x = 0:9
    x
end
```

C++

```c++
for (int x=0; x < 10; x++) {
    cout << x << endl;
}
```
For Loops

for (int x=0; x<10; x++)
{
    cout << x << endl;
}

Declaration
Loop condition
Iteration expression
For Loops

```
for (int x=0; x<10; x++)
{
    cout << x << endl;
}
```

```
int x = 0;
loop while x < 10
{
    cout << x << endl;
    x++;
}
```
For Loops

```cpp
for (int x=1; x<200; x *= 2) {
    cout << x << endl;
}
```
For Loops

```cpp
for (int x=0; x<200; x *= 2)
{
    cout << x << endl;
}
```

Press Control+C to stop an infinite loop
For Loops

```c
int main(int argc, char* argv[]) {
  for (int x=1; x<200; x *= 2) {
    // Do nothing
  }
  cout << "The largest power of two "
       << "smaller than 200 is " << x
       << endl;
  return 0;
}
```
Scope

```c
int main(int argc, char* argv[]) {
    for (int x=1; x<200; x *= 2) {
        // Do nothing
    }
    cout << "The largest power of two smaller than 200 is " << x << endl;
    return 0;
}
```

x comes into existence →

x goes out of existence →


Scope

• The **scope** of a variable is the range of code where the variable exists
• A **block** of code is a chunk of code inside of curly braces
• Variables' scope begins when they are declared, and ends when the innermost block they are declared within is closed
Scope

```cpp
int main(int argc, char* argv[]) {
    for (int x=1; x<200; x *= 2) {
        // Do nothing
    }
    cout << "The largest power of two smaller than 200 is " << x << endl;
    return 0;
}
```
int main(int argc, char* argv[]) {
    int x;
    for (x=1; x<200; x *= 2) {
        // Do nothing
    }
    cout << "The largest power of two "
    << "smaller than 200 is " << x
    << endl;
    return 0;
}
int main(int argc, char* argv[]) 
{
    int x = 1;
    for (; x<200; x *= 2) 
    {
        // Do nothing
    }
    cout << "The largest power of two "
         << "smaller than 200 is " << x
         << endl;
    return 0;
}
For Loops

```c
int main(int argc, char* argv[]) {
    int x = 1;
    for (; x<200; x *= 2) {
        // Do nothing
    }
    cout << "The largest power of two "
        << "smaller than 200 is " << x
        << endl;
    return 0;
}
```
For Loops

• Inside of a for loop, any of the components may be blank
• A common way to write an infinite loop is as `for (;;)`
• It is still possible to get out of an infinite loop

for (declaration; condition; iteration)
For Loops

• The continue statement skips to the next loop iteration

```cpp
for (int x=0; x<10; x++) {
    if (x % 2 == 0) {
        continue;
    }
    cout << x << endl;
}
```
For Loops

• The continue statement skips to the next loop iteration
• The break statement stop loop iteration completely
  • Essentially "return"ing from a loop to the rest of the function

```c
for (int x=0; x<10; x++) {
  if (x % 3 == 0) {
    break;
  }
  cout << x << endl;
}
```
For Loops Summary

• C++ for loops are written with three clauses
• Usually, you will start a block after a for-loop statement
• Variables declared in the declaration are only declared for the body of the for loop
• Individual components of the for construct can be omitted
• The break statement stops the loop
• The continue statement skips to the next loop iteration

for (declaration; condition; iteration)
Input

• In C++, simple keyboard input is simple
• Similar to cout, cin allows input
• Error handling is complicated

```cpp
cout << "Enter a number:";
int x;
cin >> x;
cout << "You entered " << x << endl;
```
```cpp
int (i=0; i<5; i++) {
    cout << "Enter a number:";
    int x;
    cin >> x;
    cout << "You entered " << x << endl;
}
```

Enter a number: 10
You entered 10
Enter a number: 20
You entered 20
Enter a number: 30
You entered 30
Enter a number: 40
You entered 40
Enter a number: 50
You entered 50
Randomness

• C++ has a random number generator that it inherited from the C programming language

• There are two key functions:
  • srand: "seed" the random number generator
  • rand: return a random number between zero and RAND_MAX

• Starting from the same seed (on the same compiler) will result in the same sequence of random numbers
Randomness

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main(int argc, char* argv[]) {
    srand(time(0));
    for (int i=0; i<5; i++) {
        cout << "Random number is: " << rand() << endl;
    }
    return 0;
}
```
Randomness

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main(int argc, char* argv[]) {
    srand(time(0));
    for (int i=0; i<5; i++) {
        cout << "Random number is: " << rand() % 20 << endl;
    }
    return 0;
}
```

Random number is: 8
Random number is: 5
Random number is: 1
Random number is: 2
Random number is: 17
Randomness – Modulus Operator

• The modulus operator is written as the percent sign (%)
• It performs integer division and returns the remainder
• This can be useful to limit integers to within a range, or to test for divisibility by a number
• For a given value of $n$, $x \% n$ will return numbers between 0 and $n-1$.
• `rand() \% N` will return a random integer between 0 and $N-1$

<table>
<thead>
<tr>
<th>x</th>
<th>$x % 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>
First Homework Assignment! Yay!

• On the website, there is a reading and programming assignment
• You will be programming a number guessing game
• This is due a week from today at the beginning of class
• You are encouraged to use reference websites to figure out how C++ works
• You must NOT copy the code for this assignment from the internet
C++ reference
C++98, C++03, C++11, C++14

Strings library
- basic_string
- Null-terminated byte strings
- Null-terminated multibyte strings
- Null-terminated wide strings

Containers library
- array (C++11)
- vector - deque
- list - forward_list (C++11)
- set - multiset
- map - multimap
- unordered_set (C++11)
- unordered_multiset (C++11)
- unordered_map (C++11)
- unordered_multimap (C++11)
- stack - queue - priority_queue

Algorithms library

Iterators library

Numerics library
- Common mathematical functions
- Complex numbers
- Pseudo-random number generation

Input/output library
- basic_streambuf
- basic_filebuf
- basic_stringbuf
- ios base
- basic_ios
- basic_istream
- basic_ostream
- basic_iostream
- basic_ifstream
- basic_ofstream
- basic_fstream
- basic_ifstream
- basic_ostream
- basic_iostream
- I/O manipulators
- C-style I/O

Localizations library

Regular expressions library (C++11)

Atomic operations library (C++11)

Thread support library (C++11)

Technical specifications

Standard library extensions (library fundamentals T5)
- optional — any — basic_string_view
- polymorphic allocators — search

Standard library extensions v2 (library fundamentals T5 v2)
- propagate_const — not_fn — observer_ptr
- source_location — ostream_joiner
- detection idioms — uniform container erasure

Filesystem library (filesystem T5)

Parallelism library extensions (parallelism T5)

Concurrency library extensions (concurrency T5)

Concepts (concepts T5)

External Links — Non-ANSI/ISO Libraries — Index