# CSCI 390 – Special Topics in C++

Lecture 9

9/18/18

Time To Turn Off Cell Phones

## C++ Lambda Functions
## Sneaking UP

- C++ lambda functions are functions with a specialized syntax that allow them to be defined “on the fly”.
  - No prototype.  No name.
  - Return type inferred from the return statement.
  - Defined where you need it.
  - Most often passed a parameter, especially for functions in <algorithm>.  (It does not need to be passed as a parameter.)
- Unusual syntax can be a bit confusing.

# C++ Lambda Functions (cont) Syntax

- Syntax:
  **[**<capture>**]** **(**<formal parameters>**)** **{**<body>**}**
- <capture> discussed later. We aren't ready for that now.
- Note there is no return type.
- <formal parameters> just like regular functions.
- <body> just like regular function, but must return a value.

## C++ Lambda Functions (cont) Simple Example

```
#include <iostream>

int main()
{
  std::cout << [](void){return "Hello World";} <<
               std::endl;

  std::cout << [](void){return "Hello World";}() <<
               std::endl;

  return 0;
}
```

```
1
Hello World


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

- **`[](void){return "Hello World";}`** is a lambda function with no parameters that returns the **`const char *`**, “Hello World”.
- The **`()`** invokes the lambda function, just like any other function.

# C++ Lambda Functions (cont) Example

```
#include <iostream>

int main()
{
  const char * (*f)(void) {[](void){return "Hello World";}};

  std::cout << f() << std::endl;

  return 0;
}
```

```
Hello World


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

- **`[](void){return "Hello World";}`** really is function with no parameters that returns the **`const char *`**, “Hello World”.

# C++ Lambda Functions (cont) Example

```
#include <iostream>

typedef const char * (*tSimpleFunc)(void);

int main()
{
  tSimpleFunc f = [](void){return "Hello World";};

  std::cout << f() << std::endl;

  return 0;
}
```

```
Hello World


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

- **`[](void){return "Hello World";}`** no matter how you cut it, it really is a function with no parameters that returns the **`const char *`**, “Hello World”.

# C++ Lambda Functions (cont) Example

```
#include <iostream>

void TestFunc(const char * (*f)(void))
{
  std::cout << "TestFunc: " << f() << std::endl;
  return;
}

int main()
{
  TestFunc([](void){return "Hello World";});
  TestFunc([](void){return "Goodbye World";});

  return 0;
}
```

```
TestFunc: Hello World
TestFunc: Goodbye World


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

- You can pass lambda functions as a parameter – just like any other function.

## C++ Lambda Functions (cont) Example

```
#include <iostream>
#include <cstdint>

void TestFunc(uint32_t (*f)(uint32_t x))
{
  std::cout << "f(2u): " << f(2u) << std::endl;
  return;
}

int main()
{
  TestFunc([](uint32_t x){return 2u * x;});
  TestFunc([](uint32_t x){return x - 1u;});

  return 0;
}
```

```
f(2u): 4
f(2u): 1


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

- Lambda functions can have parameters – just like any other function.

## Midterm

- Newton's Method
  - Practice lambda functions
  - Due Friday, 10/5/18 by 11:59:59PM
  - Unlimited submissions
  - Counts for 30% of midterm grade
    - Homeworks are the other 70%
  - Counts for 10% of final grade

# Midterm
# Sample Console Output

```
Console:
Iteration: 1, xn: 1.5, Tolerance: 0.5
Iteration: 2, xn: 1.4166666666666666666305265942505, Tolerance: 0.083333333333333333369473405749517
Iteration: 3, xn: 1.4142156862745098039194427408383, Tolerance: 0.0024509803921568627110838534122195
Iteration: 4, xn: 1.4142135623746899105931951190485, Tolerance: 2.1238998198933262476217898040431e-06
Iteration: 5, xn: 1.4142135623730950487637880730318, Tolerance: 1.5948618294070460166267366730608e-12
Iteration: 6, xn: 1.4142135623730950487637880730318, Tolerance: 0
    std::sqrt(2): 1.4142135623730951454746218587388
  std::sqrt(2)^2: 2.0000000000000004440892098500626
 Newton returned: 1.4142135623730950487637880730318
  Newton squared: 1.9999999999999999998915797827514
```

## **while** Loops

- Syntax:
  ```
  while (<expression>) <statement>;
  ```
  or, more typically:
  ```
  while (<expression>)
  {
    <statement>;
    …
  }
  ```
- Semantics: Evaluate <expression> and if non-zero execute <statement>s. Repeat until <expression> is zero.
- <statement>s may never be executed.

## **`while`** Loops (cont) Example

```cpp
#include <iostream>

int main()
{
  auto i = 3u;
  while (i)
  {
    std::cout << "i: " << i-- << std::endl;
  }

  return 0;
}
```

```
i: 3
i: 2
i: 1


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

# **`do while`** Loops

- Syntax:
  **do** <statement> **`while(`**`<expression>`**`);`**
  or, more typically:
  **do**
  **{**
    <statement>**;**
    …
  **} while (**<expression>**)**
- Semantics: Execute <statement>s. Repeat while <expression> is non-zero.
- <statement>s always executed at least once.

## **do while** Loops (cont) Example

```
#include <iostream>

int main()
{
  auto i = 0u;
  do
  {
    std::cout << "i: " << i << std::endl;
  } while(i);

  return 0;
}
```

```
i: 0


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

## **for** Loops

- Syntax:

```
for(<init stmt>; <test exp>;  <inc exp>)
<stmt>;
```

or, more typically:

```
for(<init stmt>; <test exp>;  <inc exp>)
{
  <statement>;
  …
}
```

# **`for`** Loops (cont)

- Semantics: Almost the same as:

```
<init stmt>;
while(<test stmt>)
{
    <statement>;
    …
    <inc stmt>;
}
```

- <statement>s may never be executions.

## **`for`** Loops (cont) Infinite Loops

- Syntax:

```
for( ; ; )
{
    <statement>;
    …
}
```

- Must us a **`break`** statement to exit loop. Will be covered soon.