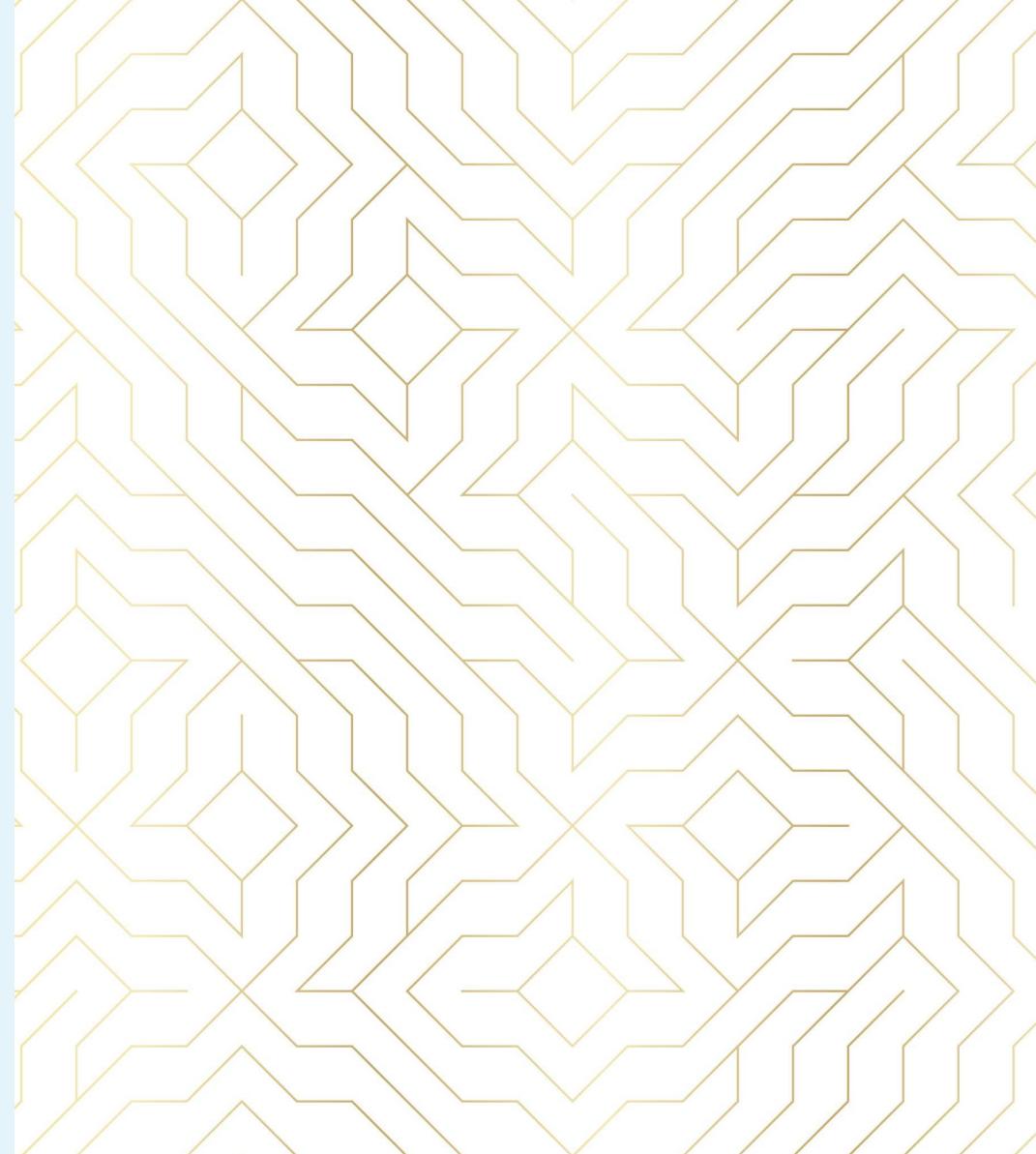


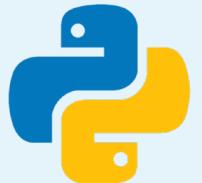
Introduction to C++ Loops

Jeremy Castagno



About Me

- Chemical Engineering and Computer Science @ Brigham Young University
- 3 years as a control systems engineer @ Valero
- 5th Year PhD Robotics @ University of Michigan
- Research focused on urgent landing for Unmanned Aerial Vehicles
- Love C++ and Python!



Getting Started

- Entire lecture and source code are hosted on Github - <https://github.com/JeremyBYU/LoopIt>
- Examples can be run in your browser using [Repl.it](#), C++ compiler and editor

The screenshot shows a Repl.it interface. On the left is a file explorer with files like cmake-build, Lecture, and src (containing For1.cpp, For2.cpp, For3.cpp, For4.cpp, StartUp.cpp, While1.cpp, While2.cpp). The main area is a code editor for src/For1.cpp:

```
src/For1.cpp
1 #include <iostream>
2 #include <string>
3 #include <vector>
4
5 int main()
6 {
7     // list of all reindeers
8     std::vector<std::string> reindeers{"Dasher", "Dancer", "Prancer",
9                                         "Vixen", "Comet", "Cupid",
10                                        "Donner", "Blitzen",
11                                        "Rudolph"};
12
13     // initialize conditional increment
14     //   |           |
15     //   v           v
16     for (int i = 0; i < reindeers.size(); ++i)
17     { // Begin Loop Body
18         std::string reindeer = reindeers[i]; // use 'i' variable
19         std::cout << "Reindeer #" << std::to_string(i) <<
20                  " is " << reindeer << std::endl;
21     } // End Loop Body
```

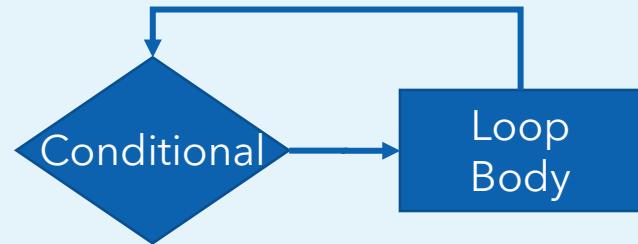
To the right is a terminal window showing the output of the program:

```
./cmake-build/bin/For1
Reindeer #0 is Dasher
Reindeer #1 is Dancer
Reindeer #2 is Prancer
Reindeer #3 is Vixen
Reindeer #4 is Comet
Reindeer #5 is Cupid
Reindeer #6 is Donner
Reindeer #7 is Blitzen
Reindeer #8 is Rudolph
Finished Loops!
```

- Lecture assumes students have a basic knowledge of basic C++ datatypes and declaring variables
 - `bool, int, string, vector`

What We Are Learning - Loops

- Often need to repeat a body of code when programming
- C++ has two main types of loops: **for** & **while**



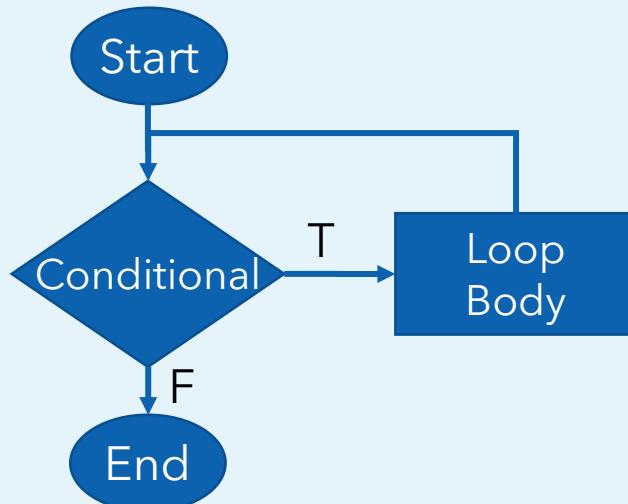
- Keywords we will be learning: **for**, **while**, **do**, **break**, **continue**
- Will *not* learn about **goto** flow control
- At the end we will learn about modern C++ **for** loops
- Examples are not meant to be *optimal or efficient*; demonstrate principles

Q: What is a Conditional?

A: Something that evaluates to either **true** (T) or **false** (F)

While Loops - Basic

- Loop forever until a condition is met



Condition Variable

while loop Statement

Loop Body

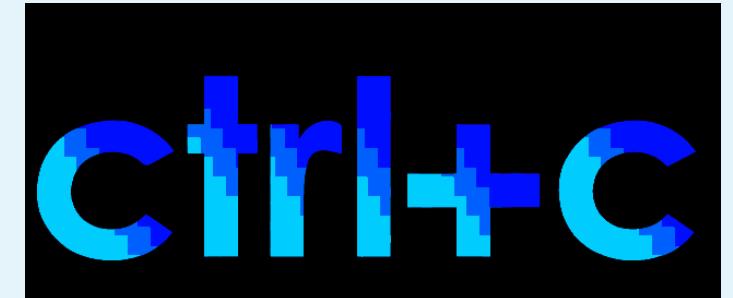
Update Variable

```
src/While1.cpp
1  #include <iostream>
2  #include <string>
3  int main()
4  {
5
6  bool repeat = true; // condition variable
7  std::string input; // user input variable
8
9  // conditional
10 // |
11 //
12 while(repeat)
13 { // Begin Loop Body
14     std::cout << "Loop execututed. Repeat? (y/n): ";
15     std::getline (std::cin, input);
16     repeat = input == "y";
17 } // End Loop Body
18 std::cout << "Finished Loops!\n";
19 }
```

The code snippet shows a C++ program named 'While1.cpp'. It includes the `<iostream>` and `<string>` headers. The `main()` function contains a `while` loop. Inside the loop, it prints a question to the console, reads user input into the `input` variable using `std::getline`, and then checks if the input is 'y' to update the `repeat` variable. Finally, it prints a message indicating the loop has finished.

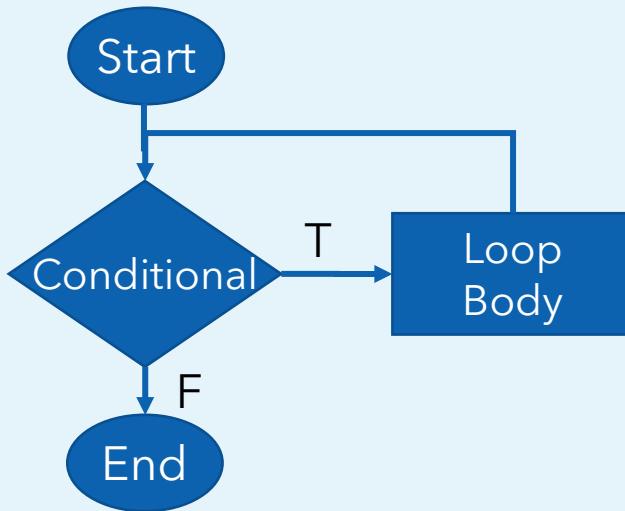
Example Program Output:

```
▶ ./cmake-build/bin/While1
Loop execututed. Repeat? (y/n): y
Loop execututed. Repeat? (y/n): y
Loop execututed. Repeat? (y/n): y
Loop execututed. Repeat? (y/n): n
Finished Loops!
```



While Loops - Break Statement

- You can also use the **break** keyword to exit a loop



Condition Statement

Exit Loop

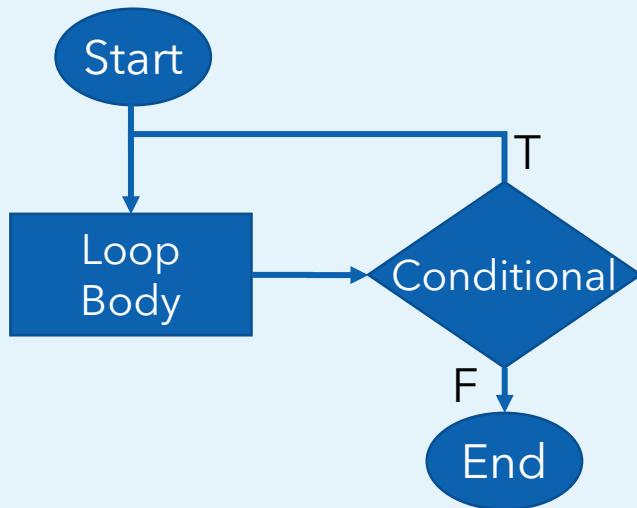
```
src/While2.cpp
1  #include <iostream>
2  #include <string>
3  int main()
4  {
5      std::string input; // user input variable
6
7      while(true)
8      { // Begin Loop Body
9          std::cout << "Loop executed. Repeat? (y/n): ";
10         std::getline (std::cin, input);
11         if (input != "y")
12         {
13             // Break out of loop
14             break;
15         }
16     } // End Loop Body
17     std::cout << "Finished Loops!\n";
18 }
19 }
```

The code snippet shows a C++ program named 'While2.cpp'. It includes the `<iostream>` and `<string>` headers. The `main()` function declares a `std::string` variable `input` for user input. A `while(true)` loop is entered. Inside the loop, it prints a prompt asking if the user wants to repeat. It reads the user's input using `std::getline`. If the input is not "y", it executes a `break` statement, which exits the loop. Finally, it prints "Finished Loops!" and ends the program.

- When to use **while** loops?
 - Don't know how many times to repeat code
 - Infinitely repeating until user input exit
 - Often used for game loops: Get Input, Update Game, Render

While Loops - Do While

- Do while loops execute Loop Body *first*, then conditional



Condition Variable

do keyword

Update Variable

Condition Statement

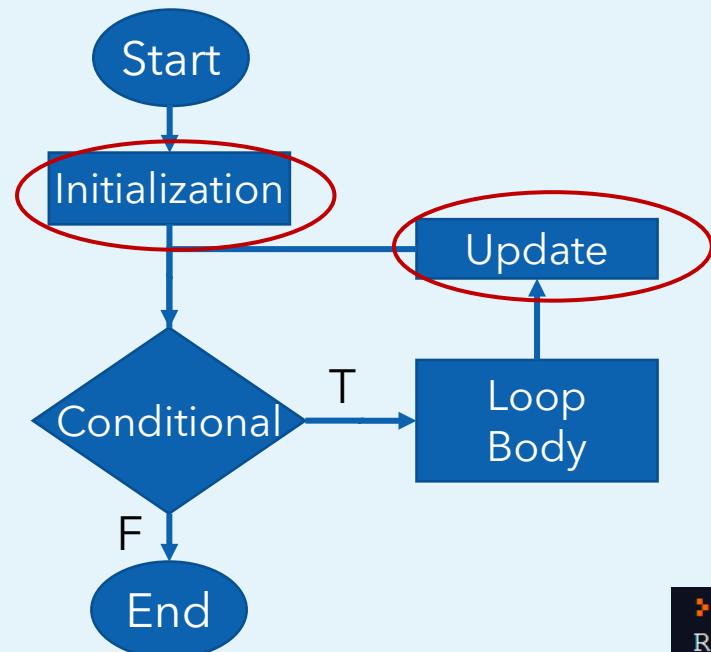
```
src/While3.cpp
1  #include <iostream>
2  #include <string>
3  int main()
4  {
5
6  bool repeat = false; // condition variable
7  std::string input; // user input variable
8
9  do
10 { // Begin Loop Body
11     std::cout << "Loop executed. Repeat? (y/n): ";
12     std::getline (std::cin, input);
13     repeat = input == "y";
14 } // End Loop Body
15 while(repeat);
16 //
17 //
18 // conditional
19 std::cout << "Finished Loops!\n";
20 }
```

The code snippet shows a C++ program named 'While3.cpp'. It includes headers for iostream and string. The main function starts with a brace on line 4. Line 6 defines a boolean variable 'repeat' and initializes it to false, with a comment indicating it's a condition variable. Line 9 contains the 'do' keyword, starting the loop body. Lines 10 through 14 are part of the loop body, which includes a cout statement asking if the user wants to repeat, a getline call to read input, and an assignment statement updating the 'repeat' variable based on the input. Line 15 contains the 'while' keyword followed by the condition 'repeat'. Lines 16 through 19 are comments explaining the structure of the loop. Line 20 concludes the main function with a brace.

- It doesn't matter that repeat is set to False in Line 6
- Loop Body is executed at *least* once

For Loops - Holiday Special

- For loops have an **initialization**, **conditional**, and **update**



Example Program Output:

```
▶ ./cmake-build/bin/For1
Reindeer #0 is Dasher
Reindeer #1 is Dancer
Reindeer #2 is Prancer
Reindeer #3 is Vixen
Reindeer #4 is Comet
Reindeer #5 is Cupid
Reindeer #6 is Donner
Reindeer #7 is Blitzen
Reindeer #8 is Rudolph
Finished Loops!
```

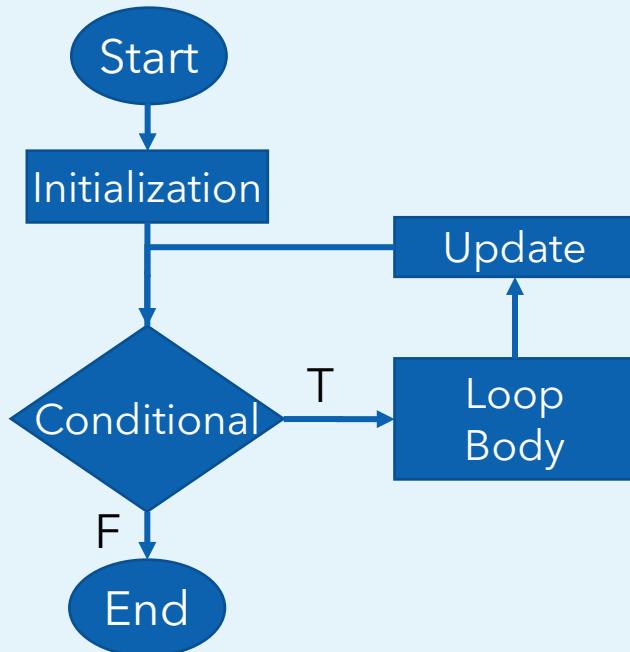
For Loop Statement
Use index

List of Reindeers

```
src/For1.cpp x
1  #include <iostream>
2  #include <string>
3  #include <vector>
4
5  int main()
6  {
7      // List of all reindeers
8      std::vector<std::string> reindeers{"Dasher", "Dancer", "Prancer",
9          "Vixen", "Comet", "Cupid",
10         "Donner", "Blitzen",
11         "Rudolph"};
12
13     // initialize    conditional    increment
14     //           ↓             ↓             ↓
15     //           |             |             |
16     for(int i = 0; i < reindeers.size(); ++i)
17     { // Begin Loop Body
18         std::string reindeer = reindeers[i]; // use 'i' variable
19         std::cout << "Reindeer #" << std::to_string(i) <<
20             " is " << reindeer << std::endl;
21     } // End Loop Body
22
23 }
```

For Loops - Continue Statement

- You can **skip** execution of remaining loop with **continue**



Conditional **in** Loop Body →

▶ ./cmake-build/bin/For2
Reindeer #0 is Dasher
Reindeer #1 is Dancer
Reindeer #2 is Prancer
Reindeer #3 is Vixen
Reindeer #4 is Comet
Reindeer #6 is Donner
Reindeer #7 is Blitzen
Reindeer #8 is Rudolph
Finished Loops!

```
5 int main()
6 {
7     // List of all reindeers
8     std::vector<std::string> reindeers{"Dasher", "Dancer", "Prancer",
9                                         "Vixen", "Comet", "Cupid",
10                                        "Donner", "Blitzen",
11                                        "Rudolph"};
12
13     // initialize      conditional      increment
14     //           |          |          |
15     //           v          v          v
16     for (int i = 0; i < reindeers.size(); ++i)
17     { // Begin Loop Body
18         std::string reindeer = reindeers[i]; // use 'i' variable
19         if (reindeer == "Cupid")
20         {
21             // Skip this reindeer, don't print name
22             continue;
23         }
24         std::cout << "Reindeer #" << std::to_string(i) <<
25             " is " << reindeer << std::endl;
26     } // End Loop Body
27     std::cout << "Finished Loops!\n";
28 }
```

The code snippet shows a C++ for loop. It initializes a vector of reindeer names. The loop iterates from index 0 to the size of the vector, printing each reindeer's name. However, if the reindeer is named "Cupid", the **continue** statement is executed, which skips the rest of the loop body for that iteration. The output shows that "Cupid" is skipped, and the program continues to print the names of the other reindeer.

Nested For Loops

- For loops can be *nested*, a loop inside a loop
- Two for loops: **outer** & **inner**
- Example printing a square
- Outer loop = rows, Inner loop = columns

	Column 0			Column 4	
Row 0	(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)
	(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
	(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
	(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)
Row 4	(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)

src/For4.cpp

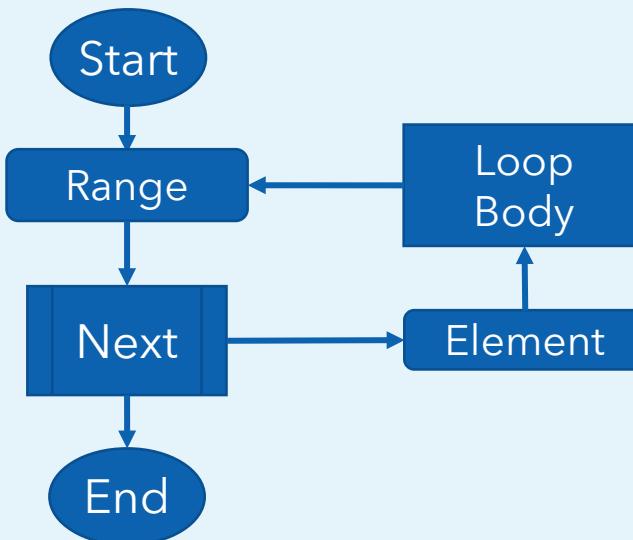
```
1 #include <iostream>
2 #include <string>
3
4 int main()
5 {
6     const int rows = 5;
7     const int cols = 5;
8     // Outer For Loop
9     for (int i = 0; i < rows; ++i)
10    {
11        // Inner for loop
12        for (int j = 0; j < cols; ++j)
13        {
14            Variable 'i' stays constant in inner loop
15            std::cout << "(" << i << "," << j << ")" << " ";
16        }
17        std::cout << std::endl;
18    }
19 }
```

Modern C++

- C++ is a language with a formal specification. However, it is a *living* language!
 - C++98, C++03, **C++11**, C++14, C++17, C++20
 - Strong backwards compatibility!
- C++11 introduced many enhancements that greatly improved code quality and efficiency
- C++11 and after is considered **Modern C++**
- We will review a new **for** loop constructs introduced in C++11: Range-based for loops
- These don't replace previously discussed patterns; they simply augment them
 - Specifically meant for ranges, aka sequences/lists

Range-based For Loop

- Execute loop body for each *element* inside *Range*
- A *Range* is a sequence of elements, e.g., [1, 2, 3]



element range(list)

Use element

- Reindeer vector is not modified/changed

src/For3.cpp

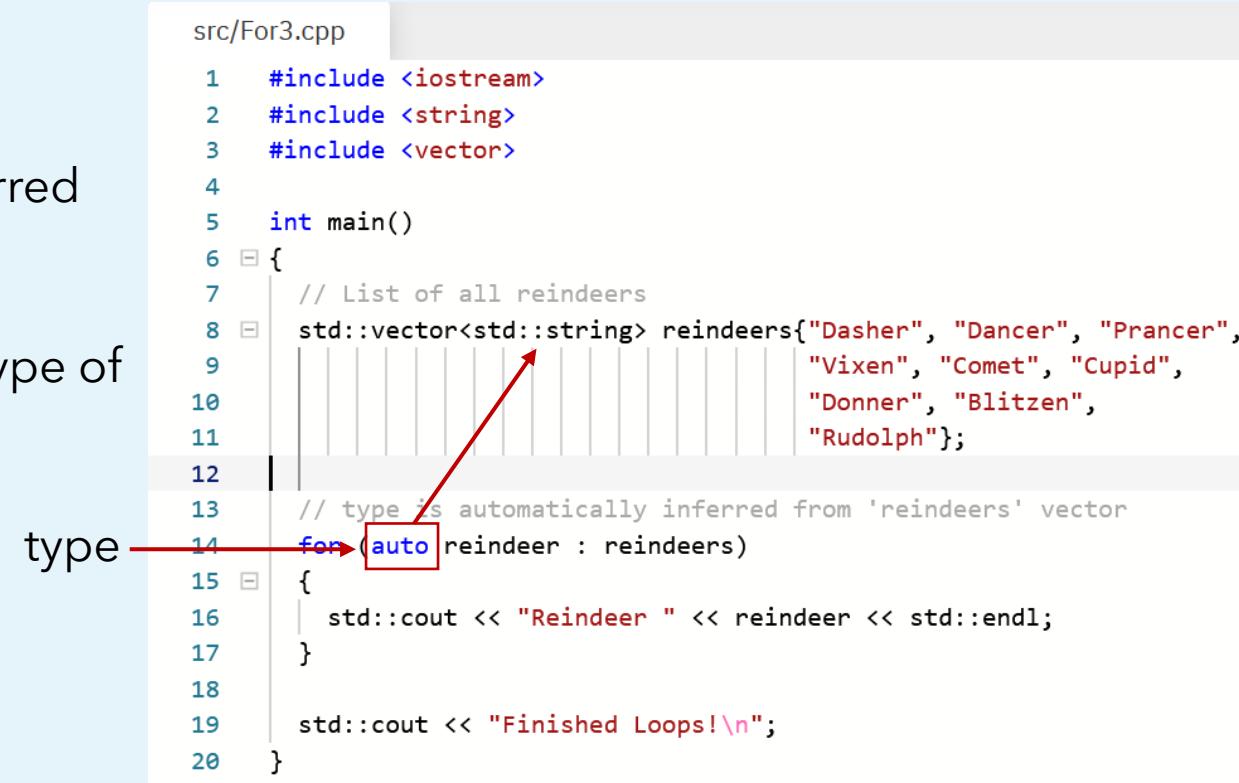
```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4
5  int main()
6  {
7      // List of all reindeers
8      std::vector<std::string> reindeers{"Dasher", "Dancer", "Prancer",
9                                         "Vixen", "Comet", "Cupid",
10                                        "Donner", "Blitzen",
11                                        "Rudolph"};
12
13     //      type      element      range (e.g. list/vector)
14     //          ↓          ↓          ↓
15     //          For (std::string reindeer : reindeers)
16     {
17         std::cout << "Reindeer " << reindeer << std::endl;
18     }
19 }
```

Example Program Output:

```
► ./cmake-build/bin/For3
Reindeer Dasher
Reindeer Dancer
Reindeer Prancer
Reindeer Vixen
Reindeer Comet
Reindeer Cupid
Reindeer Donner
Reindeer Blitzen
Reindeer Rudolph
```

Range-based For Loop - Auto!

- C++ 11 introduced the **auto** keyword
- **auto** declares a variable whose type is deduced/inferred
- **auto n = 10 ;** The variable n is of type **int**
- Range-based for loops can use auto to deduce the type of the *element* inside of the range
- This makes code less verbose



```
src/For3.cpp
1 #include <iostream>
2 #include <string>
3 #include <vector>
4
5 int main()
6 {
7     // List of all reindeers
8     std::vector<std::string> reindeers{"Dasher", "Dancer", "Prancer",
9                                         "Vixen", "Comet", "Cupid",
10                                        "Donner", "Blitzen",
11                                        "Rudolph"};
12
13 // type is automatically inferred from 'reindeers' vector
14 for (auto reindeer : reindeers)
15 {
16     std::cout << "Reindeer " << reindeer << std::endl;
17 }
18
19 std::cout << "Finished Loops!\n";
20 }
```

type → **auto**

Assignment - Count Digits

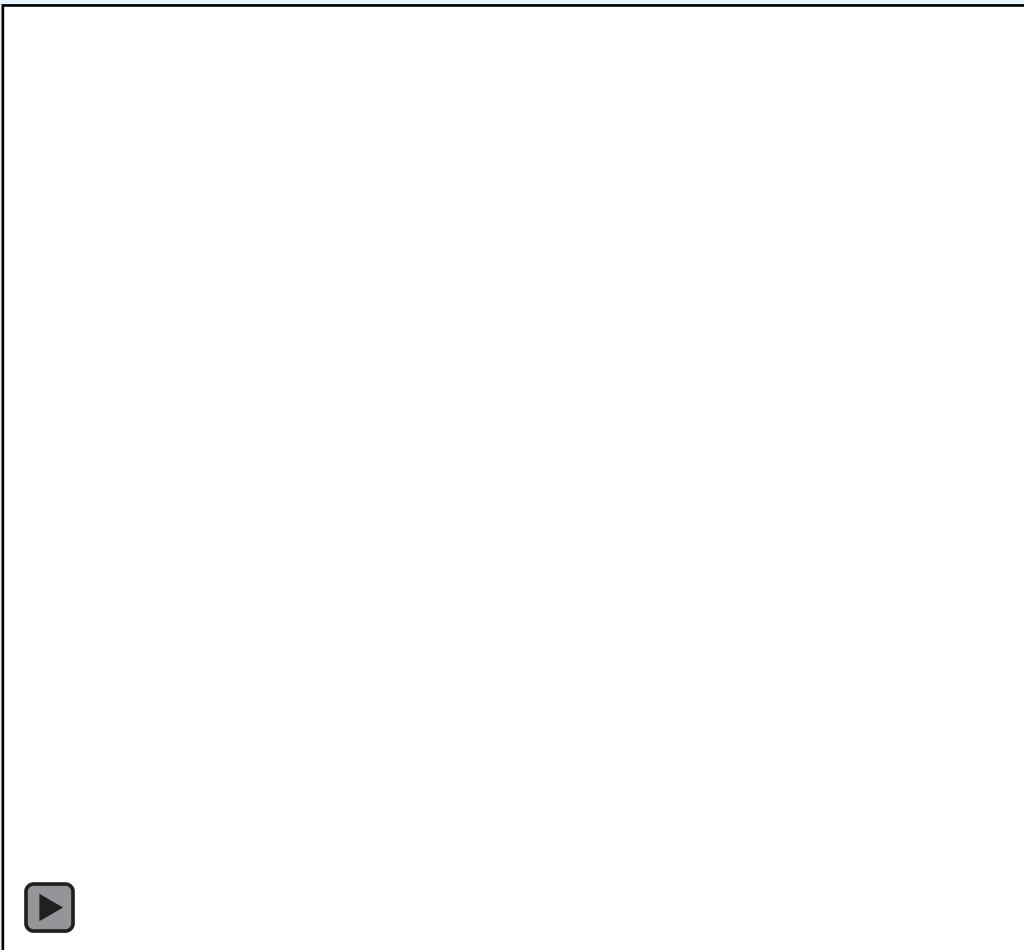
- Create a program that reads an integer and counts the number of digits; display the result
- Solve this problem using loops
- Think about which type of loop would work best in this example
- Hints: The decimal place of a number is moved left by dividing by 10

```
Jeremy@DESKTOP-QD77K4V MSYS ~/Documents/UMICH/jobs/LoopIt/cmake-build (dev)
$ ./bin/Release/CountDigit.exe
Type in your integer: 99
Total Digits: 2
(unreal)
Jeremy@DESKTOP-QD77K4V MSYS ~/Documents/UMICH/jobs/LoopIt/cmake-build (dev)
$ ./bin/Release/CountDigit.exe
Type in your integer: 88
Total Digits: 2
(unreal)
Jeremy@DESKTOP-QD77K4V MSYS ~/Documents/UMICH/jobs/LoopIt/cmake-build (dev)
$ ./bin/Release/CountDigit.exe
Type in your integer: 101
Total Digits: 3
(unreal)
```

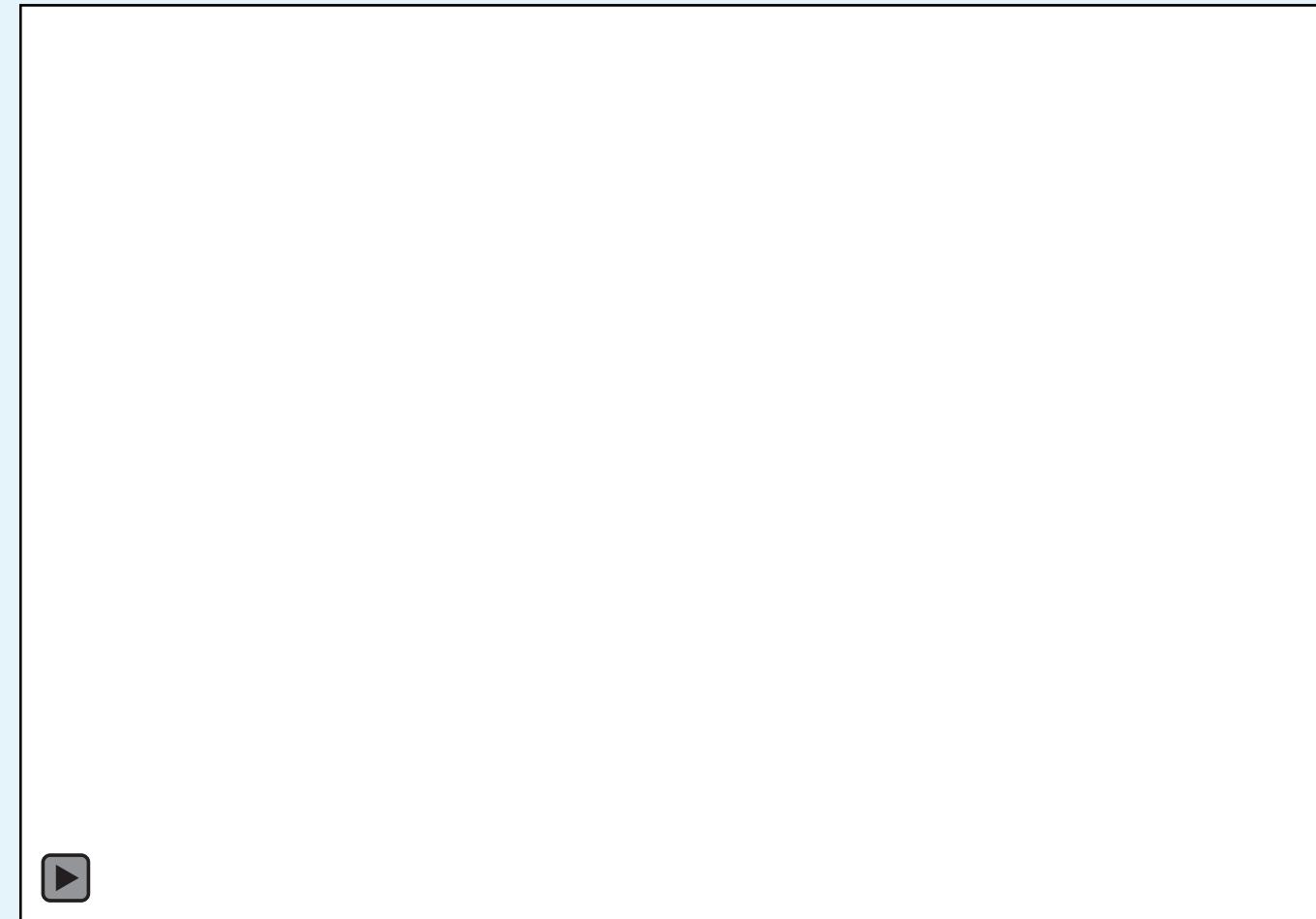
Help! My program won't stop!

If you ever run into a situation where your program won't terminate with CTRL-C

Windows - Task Manager



Linux - htop



Recap and what's next?

Recap

- Hello World
- Variables
- Conditionals
- **Loops**

What's Next

- Functions!
- Declare them and use in the same file
- Use functions defined across multiple files
 - Learn about C++ linking
- C++11 lambda, AKA anonymous functions