09 - Going Loopy - Part 2

Dr. Robert Lowe

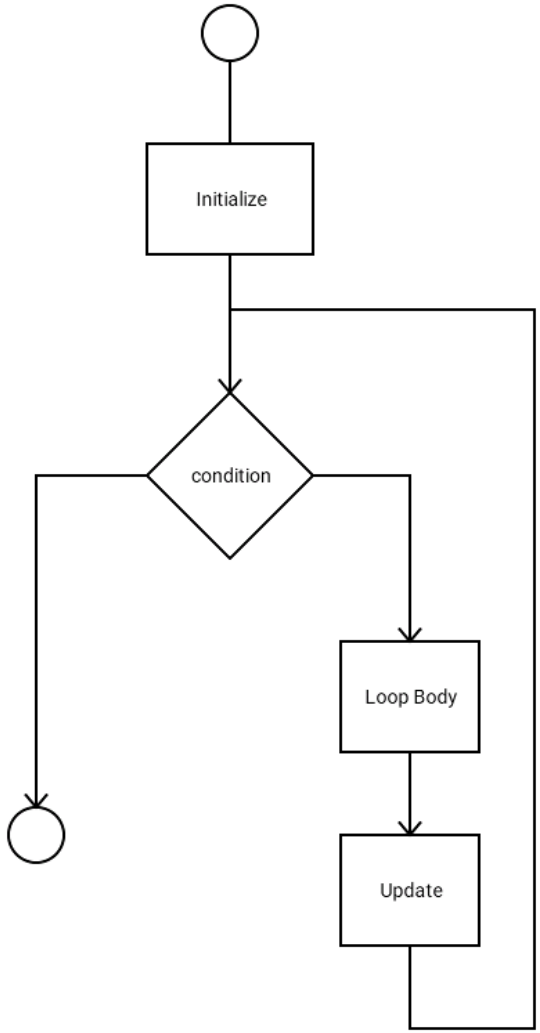
Division of Mathematics and Computer Science
Maryville College

Maryville COLLEGE

Outline

Outline

A Common Type of Loop

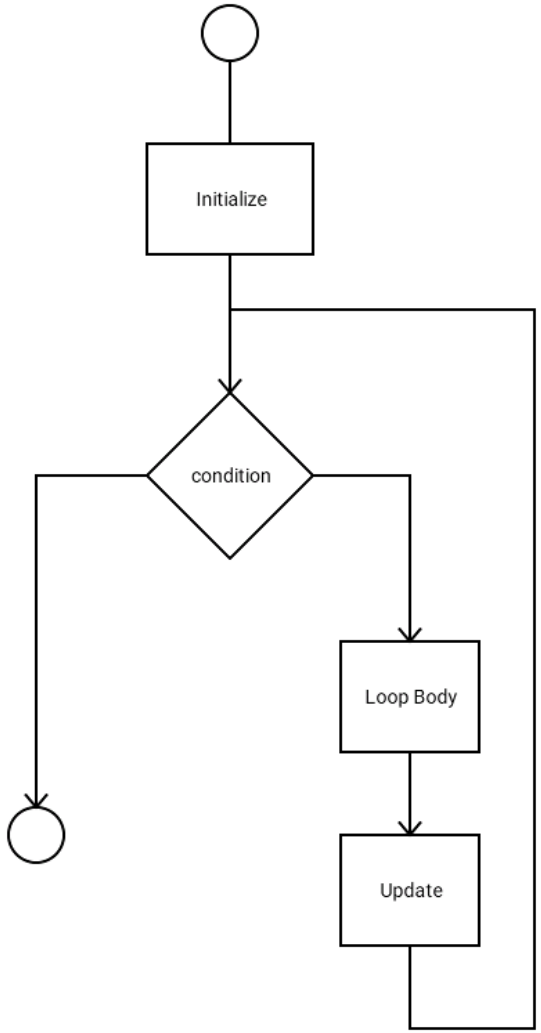


A Common Type of Loop

Basic Counting Loop

initialize

```
while(condition) {
    loop body
    update
}
```



A Common Type of Loop

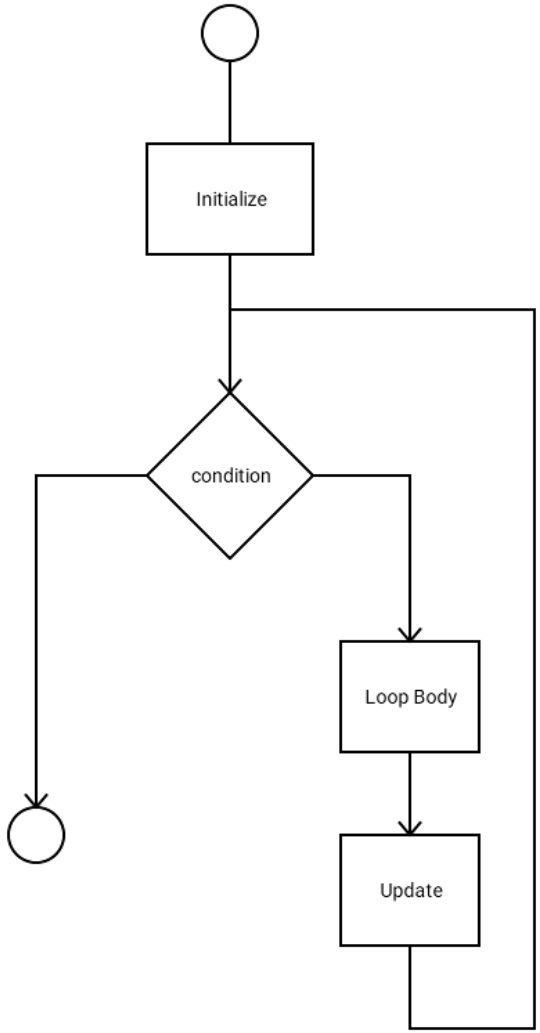
Basic Counting Loop

initialize

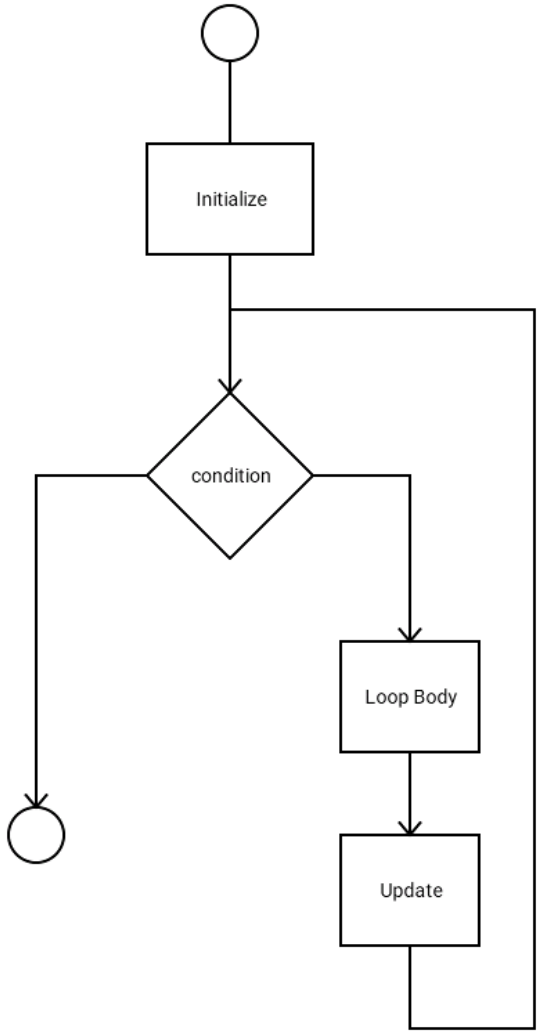
```
while(condition) {
    loop body
    update
}
```

Example: Count to 10

```
num = 0;
while(num <= 10) {
    cout << num << endl;
    num++;
}
```



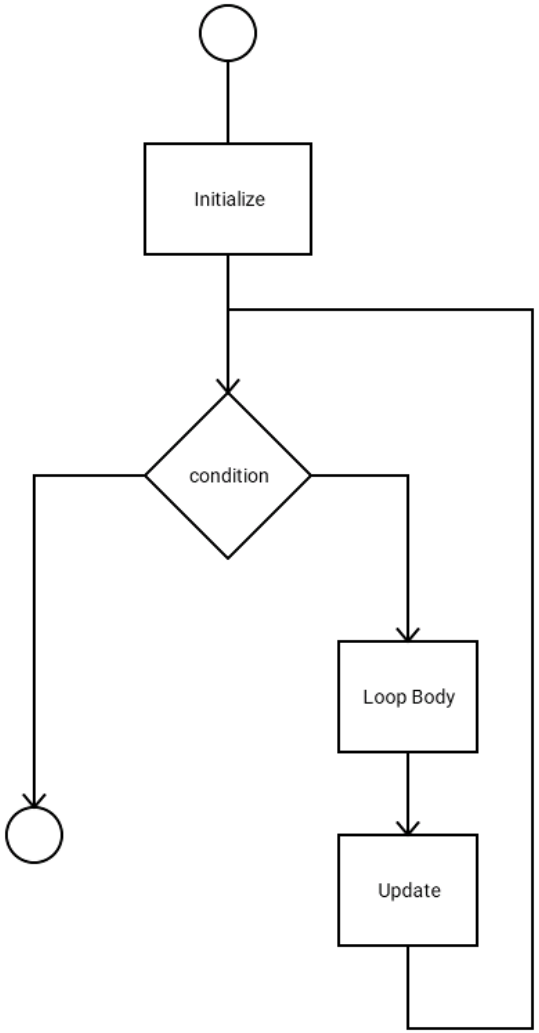
`for`: A Convenient Pre-Test Loop Format



`for`: A Convenient Pre-Test Loop Format

The For Loop

```
for( initialize; condition; update) {
    loop body
}
```



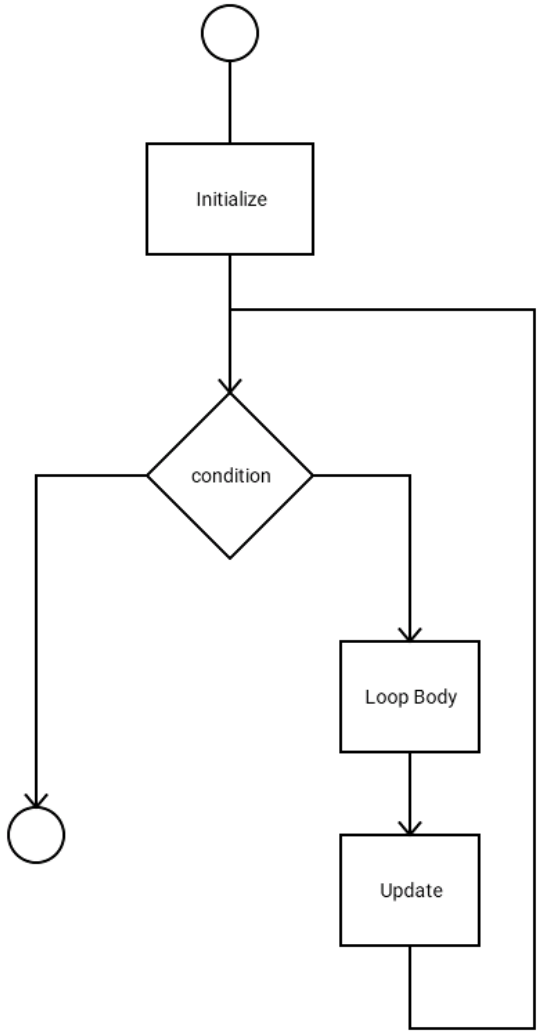
`for`: A Convenient Pre-Test Loop Format

The For Loop

```
for( initialize; condition; update) {
    loop body
}
```

Example: Count to 10

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



Declaring Variables During Loop Initialization

- If a variable is only used within the loop, it is customary to declare it in the `for` loop's initializer.

Declaring Variables During Loop Initialization

- If a variable is only used within the loop, it is customary to declare it in the `for` loop's initializer.
- Take, for example, the main function of `examples/09-Loopy/count.cpp`.

```
int main()
{
    //count to 10
    for(int num=0; num <= 10; num++) {
        //display the number
        cout << num << endl;
    }
}
```

Declaring Variables During Loop Initialization

- If a variable is only used within the loop, it is customary to declare it in the `for` loop's initializer.
- Take, for example, the main function of `examples/09-Loopy/count.cpp`.

```
int main()
{
    //count to 10
    for(int num=0; num <= 10; num++) {
        //display the number
        cout << num << endl;
    }
}
```

- Note that when you do this, the variable is **only** available in the `for` loop.

Dr. Lowe's Guide to Loop Selection

- Use a `for` loop when:

Dr. Lowe's Guide to Loop Selection

- Use a `for` loop when:
  - You are counting.

Dr. Lowe's Guide to Loop Selection

- Use a `for` loop when:
  - You are counting.
  - You have a fixed number of iterations.

Dr. Lowe's Guide to Loop Selection

- Use a `for` loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:
  - You are scanning for some sentinel data.

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:
  - You are scanning for some sentinel data.
  - You are reading to the end of input.

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:
  - You are scanning for some sentinel data.
  - You are reading to the end of input.
  - You are waiting for some general condition to be met.

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:
  - You are scanning for some sentinel data.
  - You are reading to the end of input.
  - You are waiting for some general condition to be met.
- Use a **`do..while`** loop when:

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:
  - You are scanning for some sentinel data.
  - You are reading to the end of input.
  - You are waiting for some general condition to be met.
- Use a **`do..while`** loop when:
  - You are validating user input.

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:
  - You are scanning for some sentinel data.
  - You are reading to the end of input.
  - You are waiting for some general condition to be met.
- Use a **`do..while`** loop when:
  - You are validating user input.
  - You are building a menu interface.

Dr. Lowe's Guide to Loop Selection

- Use a **`for`** loop when:
  - You are counting.
  - You have a fixed number of iterations.
  - You are exploring the entire contents of a list.
- Use a **`while`** loop when:
  - You are scanning for some sentinel data.
  - You are reading to the end of input.
  - You are waiting for some general condition to be met.
- Use a **`do..while`** loop when:
  - You are validating user input.
  - You are building a menu interface.
  - You need to go through a loop at least one time.

Converting a `while` Loop to a For Loop

```
int num;

//initialize
num = 0;

//count to 10
while(num <= 10) {
    //display the number
    cout << num << endl;

    //increment
    num++;
}
```

```
//count to 10
for(int num=0; num <= 10; num++) {
    //display the number
    cout << num << endl;
}
```

Lab Activity: Convert to For Loops

1. Create the directory `labs/week6`
2. Copy the following files from `labs/week4` to `labs/week6`.
   - `count2.cpp`
   - `fahrenheit.cpp`
3. Convert the loops in these programs to `for` loops.
4. Compile and test your programs.

Outline

Incrementing Twice

- One common mistake to make with `for` loops is to increment your counting variable twice.

Incrementing Twice

- One common mistake to make with `for` loops is to increment your counting variable twice.
- For instance, consider the following:

```
//count to 10
for(int num=0; num <= 10; num++) {
    //display the number
    cout << num << endl;

    //update the number
    num++;
}
```

Incrementing Twice

- One common mistake to make with `for` loops is to increment your counting variable twice.
- For instance, consider the following:

```
//count to 10
for(int num=0; num <= 10; num++) {
    //display the number
    cout << num << endl;

    //update the number
    num++;
}
```

- Fixing this is pretty easy, just remove the extra update!

On the Dangers of Doubles

- Change into the `examples/09-Loopy` directory.

On the Dangers of Doubles

- Change into the `examples/09-Loopy` directory.
- Compile and run the `double-count.cpp` example.

On the Dangers of Doubles

- Change into the `examples/09-Loopy` directory.
- Compile and run the `double-count.cpp` example.
- This program should count from 0 to the specified `double` using a given number of lines, each consisting of 4 columns (counting down each column).

On the Dangers of Doubles

- Change into the `examples/09-Loopy` directory.
- Compile and run the `double-count.cpp` example.
- This program should count from 0 to the specified `double` using a given number of lines, each consisting of 4 columns (counting down each column).
- Consider the following sample run:

```
What should I count to? 1.0
How many lines? 2
0.0000  0.2857  0.5714  0.8571
0.1429  0.4286  0.7143  1.0000
```

On the Dangers of Doubles

- Change into the `examples/09-Loopy` directory.
- Compile and run the `double-count.cpp` example.
- This program should count from 0 to the specified `double` using a given number of lines, each consisting of 4 columns (counting down each column).
- Consider the following sample run:

```
What should I count to? 1.0
How many lines? 2
0.0000  0.2857  0.5714  0.8571
0.1429  0.4286  0.7143  1.0000
```

- Play around with this a bit. Does it always work?

On the Dangers of Doubles

- Change into the `examples/09-Loopy` directory.
- Compile and run the `double-count.cpp` example.
- This program should count from 0 to the specified `double` using a given number of lines, each consisting of 4 columns (counting down each column).
- Consider the following sample run:

```
What should I count to? 1.0
How many lines? 2
0.0000  0.2857  0.5714  0.8571
0.1429  0.4286  0.7143  1.0000
```

- Play around with this a bit. Does it always work?
- No! This is because of the imprecision in `double` calculations and comparison!

Best Practices for Loops With Doubles

- We should generally avoid using `double` variables to perform counting.

Best Practices for Loops With Doubles

- We should generally avoid using `double` variables to perform counting.
- If we want to iterate through doubles, we should instead count using an `int`.

Best Practices for Loops With Doubles

- We should generally avoid using `double` variables to perform counting.
- If we want to iterate through doubles, we should instead count using an `int`.
- Calculate the double within the loop.

Best Practices for Loops With Doubles

- We should generally avoid using `double` variables to perform counting.
- If we want to iterate through doubles, we should instead count using an `int`.
- Calculate the double within the loop.
- This is often done using the comma operator to do two calculations in the update:

```
line++, start += increment
```

Lab Activity: Repair `double-count.cpp`

1. Copy `examples/09-Loopy/double-count.cpp` to `labs/week6`.
2. Repair the loop by changing it to this:

```
//Go through each row
double start=0.0;
for(int line=0; line < lines;  line++, start += increment) {
    //find max for this row
    double row_max = start + 3.0 * lines * increment;

    if(row_max <= max) {
        //print all the columns
        double num=start;
        for(int col=0; col<4; col++, num+=lines * increment){
            cout << num << "\t";
        }
        cout << endl;
    }
}
```

Infinite Loops and Other Pitfalls

- Another common mistake is to forget to change a loop variable. This will cause an infinite loop.

Infinite Loops and Other Pitfalls

- Another common mistake is to forget to change a loop variable. This will cause an infinite loop.
- Misplaced semicolons are also a common way to mistakenly generate an infinite loop.

```
while(x<=5); {
```

Infinite Loops and Other Pitfalls

- Another common mistake is to forget to change a loop variable. This will cause an infinite loop.
- Misplaced semicolons are also a common way to mistakenly generate an infinite loop.

```
while(x<=5); {
```

- Off by one errors are also a common mistake!

Infinite Loops and Other Pitfalls

- Another common mistake is to forget to change a loop variable. This will cause an infinite loop.
- Misplaced semicolons are also a common way to mistakenly generate an infinite loop.

```
while(x<=5); {
```

- Off by one errors are also a common mistake!
- Be careful about < vs <=, and be sure to select the correct one!