Loops

Today

- Common errors with control structures
- While loop
- do While Loop
- for loop
- for vs. while loop
- Common loop algorithms for strings

Due this week

- **Homework 3**
  - Write solutions in VSCode and paste in Autograder, **Homework 3 CodeRunner**.
  - Zip your .cppfiles and submit on canvas **Homework 3**. Check the due date! **No late submissions!!**
- Extra-credit: coderunner (start early bonus (3 points)) + coderunner (extra credit (3 points))
- Start going through the textbook readings and watch the videos
  - Take **Quiz 4**.
  - Check the due date! **No late submissions!!**
- Practice Set 3
- Week 4: 3-2-1

Practicum 1

- Coming up in week 5: Feb 7th @ 7:30 pm
- Covers material from weeks 1 – 4 and H3
    - .cppprograms
    - Variables, arithmetic, cin, cout
    - If-else, nested if-else, switch statements
- Chapters 1, 2 and 3 from the textbook (everything!)
- Two parts
    - MCQ – 4 questions
    - CodeRunner – 4 questions
- How do I prepare?
    - **Practice questions**

Practicum – Format and Rules

There are 2 parts: 75 minutes

- Multiple Choice Questions: 4 questions
- CodeRunner: 4 questions

Logistics:

- You will have access to VS Code and all your previously developed code.
- Need to join zoom meeting with video
  - Student ID: BuffOneCard, Government ID
- You have **75 minutes** to complete it
  - Focus on your own solution
  - This is an individual assessment!
  - There is not enough time to help others

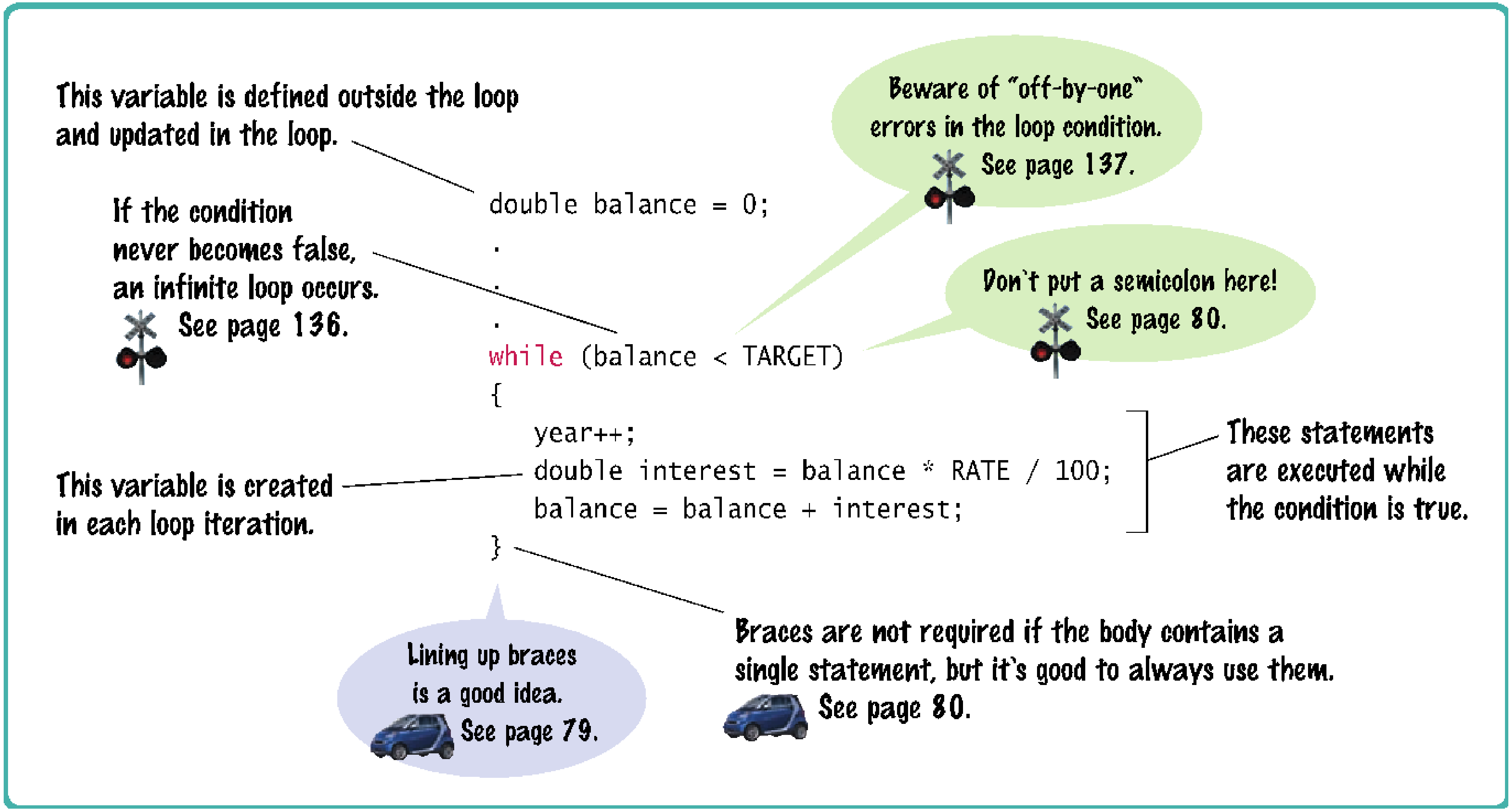
Practicum - Practice, practice, practice!

- Review all previous CodeRunner programming questions from previous homework assignments, especially H3
- Review examples we did in class
- Practice:
    - Practice Practicum MCQ
    - Practice Practicum Code Runner
- Time is short; prepare accordingly
    - Create files with placeholders ahead of time
    - Time yourself on 3 or 4 practice problems

Tips for Timed Exam

- Read the Questions
    - read them not once, but **TWICE** before starting to code
    - follow all the instructions explicitly (especially for names and order of input values from the user)
- Create or Modify a Code
    - know your C++ syntax
- Create and Use an IF, IF ELSE, SWITCH
    - know your C++ syntax
    - know how to create a condition
    - know how and when to use SWITCH. and “break;”
- Spot and fix errors!

The while Loop Syntax



This variable is defined outside the loop and updated in the loop.

If the condition never becomes false, an infinite loop occurs. See page 136.

```
double balance = 0;
.
.
.
while (balance < TARGET)
{
   year++;
   double interest = balance * RATE / 100;
   balance = balance + interest;
}
```

Beware of "off-by-one" errors in the loop condition. See page 137.

Don't put a semicolon here! See page 80.

These statements are executed while the condition is true.

This variable is created in each loop iteration.

Lining up braces is a good idea. See page 79.

Braces are not required if the body contains a single statement, but it's good to always use them. See page 80.

Example of Normal Execution

```
i = 5;
while (i > 0)
{
  cout << i << " ";
  i--;
}
```

What is the output?

Example of a Problem – An Infinite Loop

The output never ends

- *i* is set to 5
- The *i++;* statement makes *i* get bigger and bigger
- the condition will never become false
- an infinite loop

```
i = 5;
while (i > 0)
{
  cout << i << " ";
  i++;
}
```

The output is - 5 6 7 8 9 10 11

Common Error – Infinite Loops

- Forgetting to update the variable used in the condition is common.
- In the investment program, it might look like this:

```
year = 1;
while (year <= 20)
{
  balance = balance * (1 + RATE / 100);
}
```

- The variable year is not updated in the loop body!

Another Programmer Error

What is the output?

```
i = 5;
while (i < 0)
{
  cout << i << " ";
  i--;
}
```

A Very Difficult Error to find (especially after looking for hours and hours!)

What is the output?

```
i = 5;
while (i < 0)
{
  cout << i << " ";
  i--;
}
```

The do{ } while() Loop

- The while() loop's condition test is the first thing that occurs in its execution.
- The do loop (or do-while loop) has its condition tested only after at least one execution of the statements. The test is at the bottom of the loop:

```
do
{
  statements
}
while (condition);
```

The do Loop

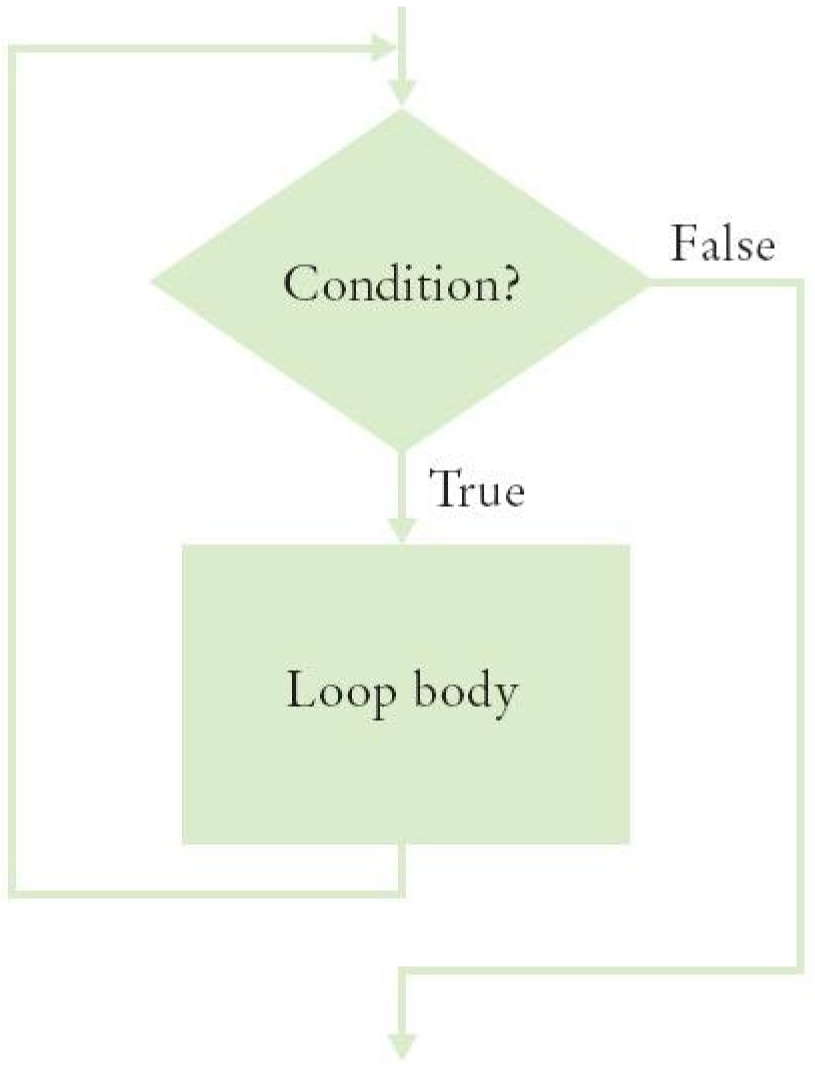
- This means that the do loop should be used only when the statements must be executed before there is any knowledge of the condition.
- This also means that the do loop is the least used loop.

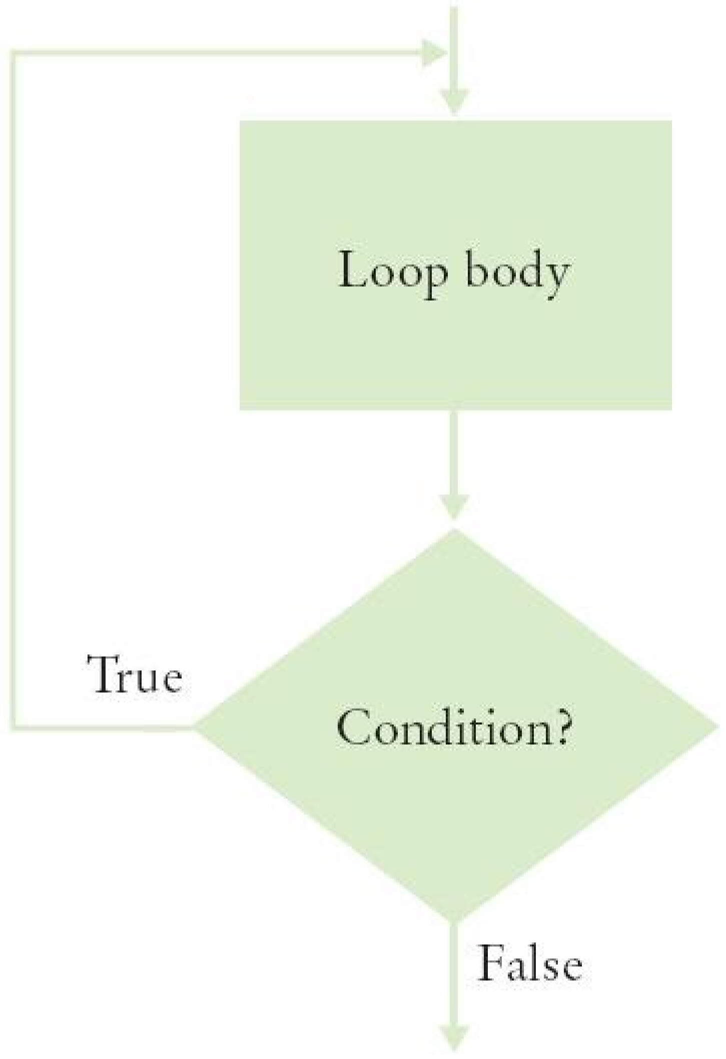
do{ } Loop Code: getting user input Repeatedly

- Code to keep asking a user for input until it satisfies a condition, such as non-negative for applying the sqrt():

```
double value;
do
{
  cout << "Enter a number >= 0: ";
  cin >> value;
}
while (value < 0);
cout << "The square root is " << sqrt(value) << endl;
```

Flowcharts for the while Loop and the do Loop





Practice It: Example of do...while

- What output does this loop generate?

```
do
{
  int value = j * 2;
  j++;
  cout << value << ", ";
} while (j <= 5);
```

How to Write a Loop

These are the steps to follow when turning a problem description into a code loop:

1. Decide what work must be done inside the loop
    - *For example, read another item or update a total*
2. Specify the loop condition
    - *Such as exhausting a count or invalid input*
3. Determine the loop type
    - *Use for in counting loops, while for event-controlled*
4. Set up variables for entering the loop for the first time
5. Process the result after the loop has finished
6. Trace the loop with typical examples
7. Implement the loop in C++

The for Loop vs. the while loop

- Often you will need to execute a sequence of statements a given number of times.

You could use a whileloop:

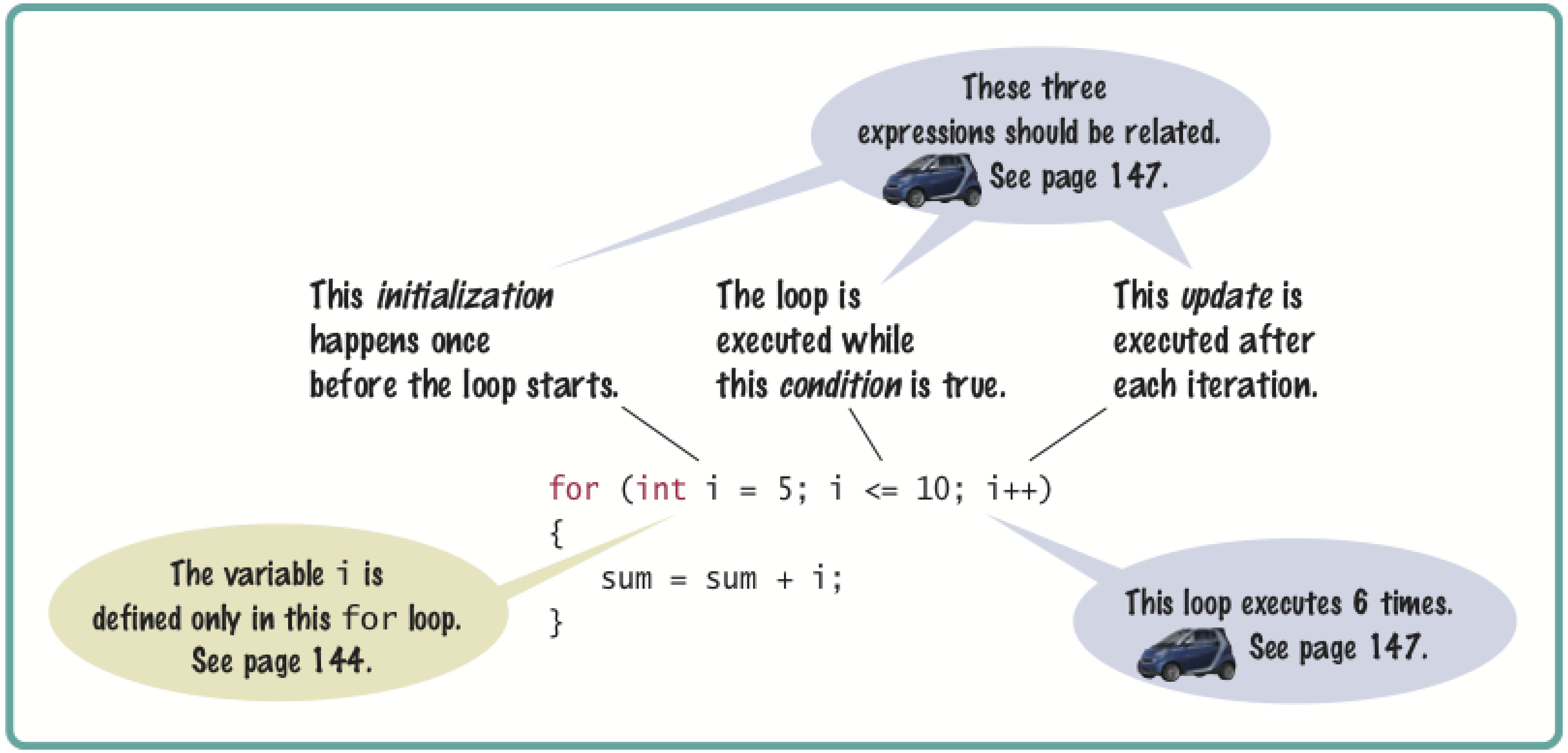
```
num = 1; // Initialize the variable
while (num <= 10) // Check the variable
{
  cout << num << endl;
  num++; // Update the variable
}
```

The for Loop

- C++ has a statement custom made ***for*** this sort of processing: the **for** loop.

```
for (num = 1; num <= 10; num++)
{
  cout << num << endl;
}
```

The for Loop Syntax

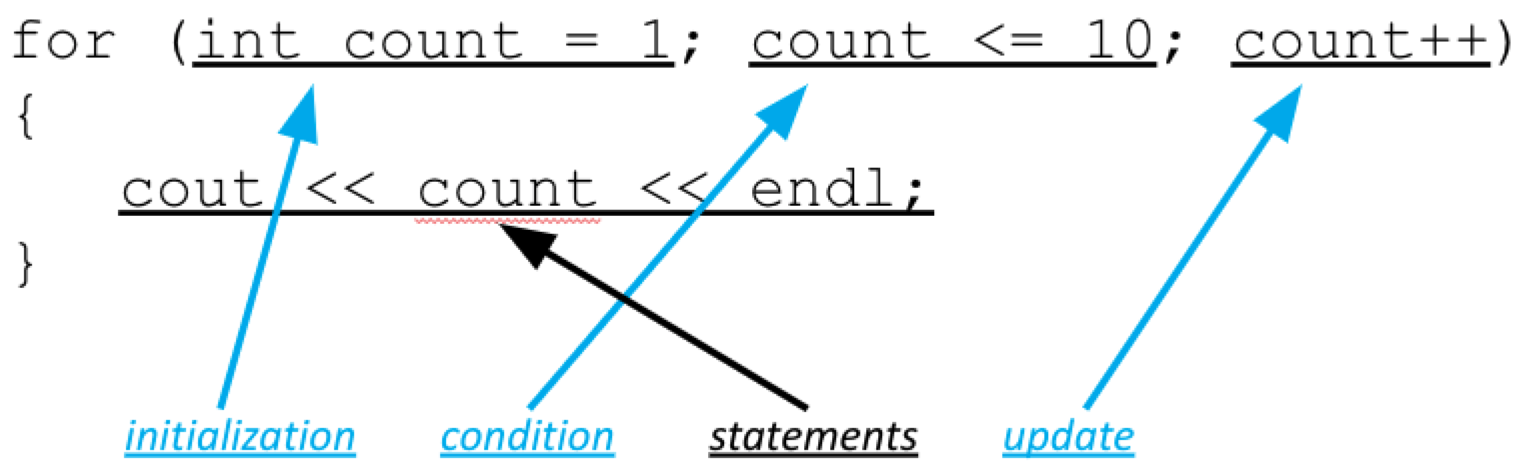


```
for (int i = 5; i <= 10; i++)
{
   sum = sum + i;
}
```

These three expressions should be related. See page 147.
This *initialization* happens once before the loop starts.
The loop is executed while this *condition* is true.
This *update* is executed after each iteration.
The variable i is defined only in this for loop. See page 144.
This loop executes 6 times. See page 147.

The for Loop Is Better than while for Certain Things

- Doing something a known number of times or causing a variable to take on a sequence of values is so common, C++ has a statement just for that:



```
for (int count = 1; count <= 10; count++)
{
    cout << count << endl;
}
```

for() loop execution

```
for (initialization; condition; update)
{
  statements;
}
```

- The ***initialization*** is code that happens once, before the check is made, to set up counting how many times the *statements* will happen. The loop variable may be created here, or before the for() statement.
- The ***condition*** is a comparison to test if the loop is done. When this test is false, we skip out of the for(), going on to the next statement.
- The ***update*** is code that is executed at the bottom of each iteration of the loop, immediate before re-testing the condition. Usually it is a counter increment or decrement.
- The ***statements*** are repeatedly executed until the condition is false. These also are known as the "loop body".

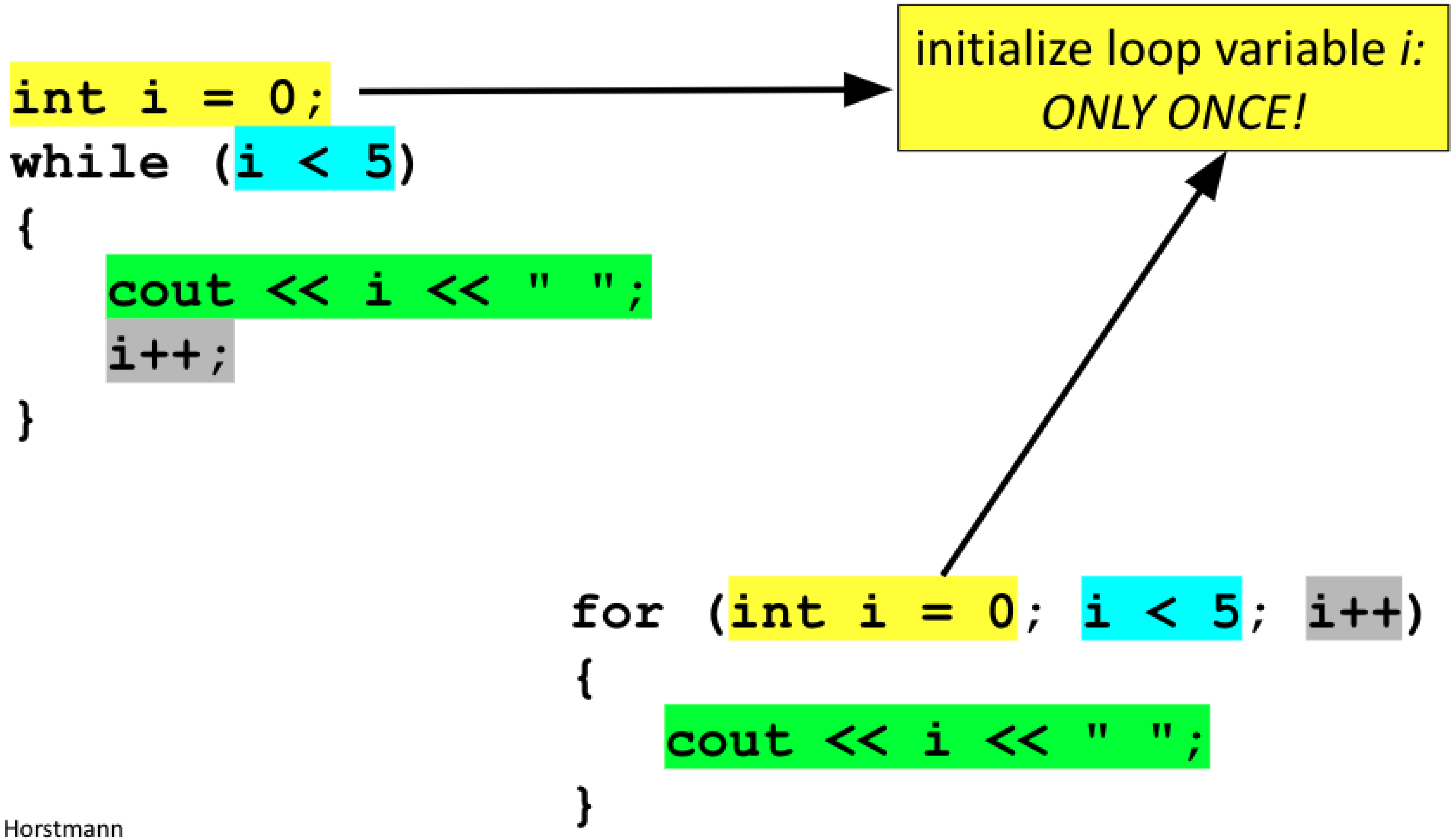
The for Can Count Up or Down

- A for loop can count down instead of up:

```
for (int counter = 10; counter >= 0; counter--)...
```

- Notice that in this examples, the loop variable is defined **in** the initialization (where it really should be!).

Converting from a while loop to a for loop

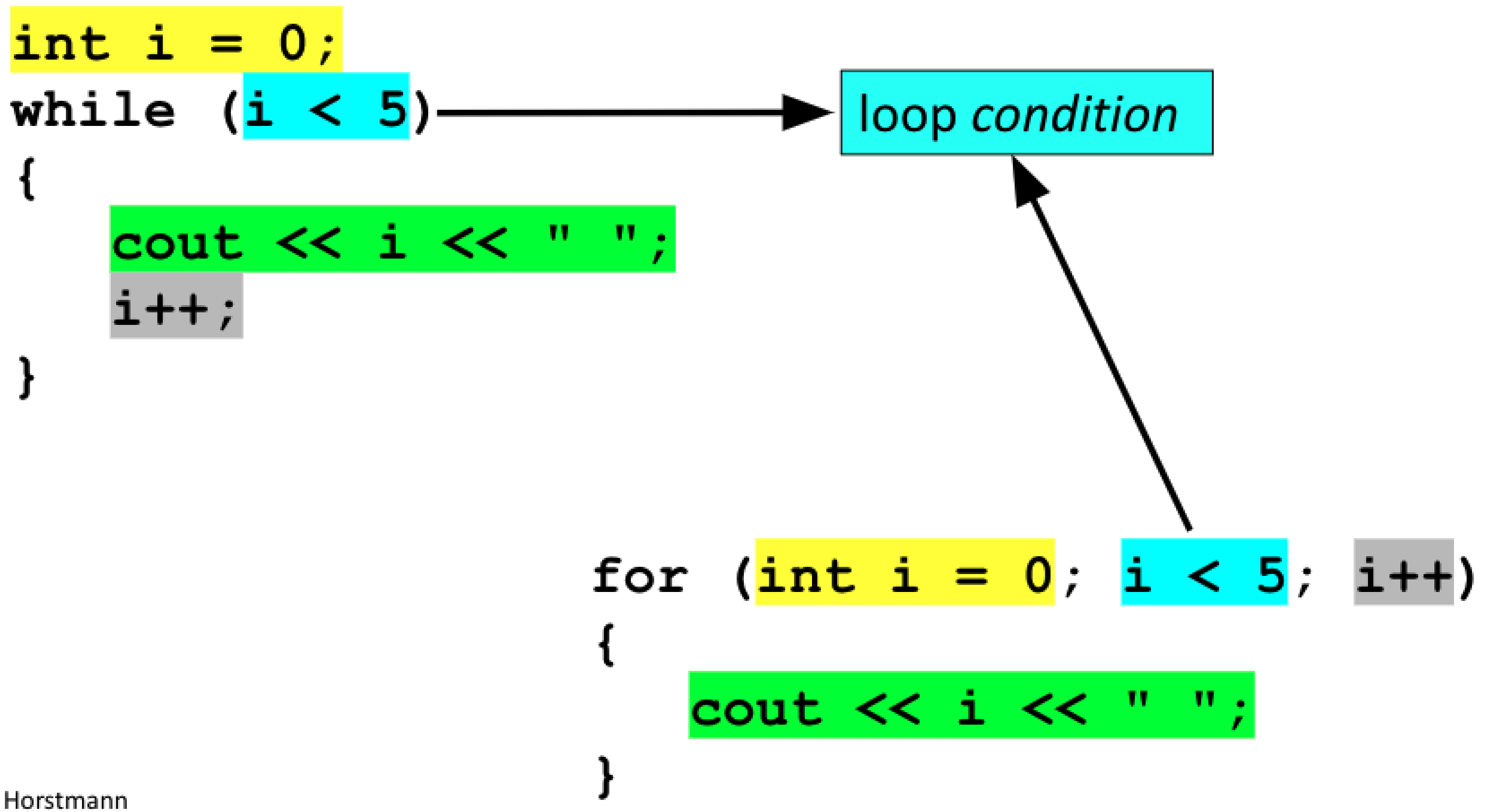


```
int i = 0;
while (i < 5)
{
    cout << i << " ";
    i++;
}
```

initialize loop variable *i:*
ONLY ONCE!

```
for (int i = 0; i < 5; i++)
{
    cout << i << " ";
}
```

Converting from a while loop to a for loop

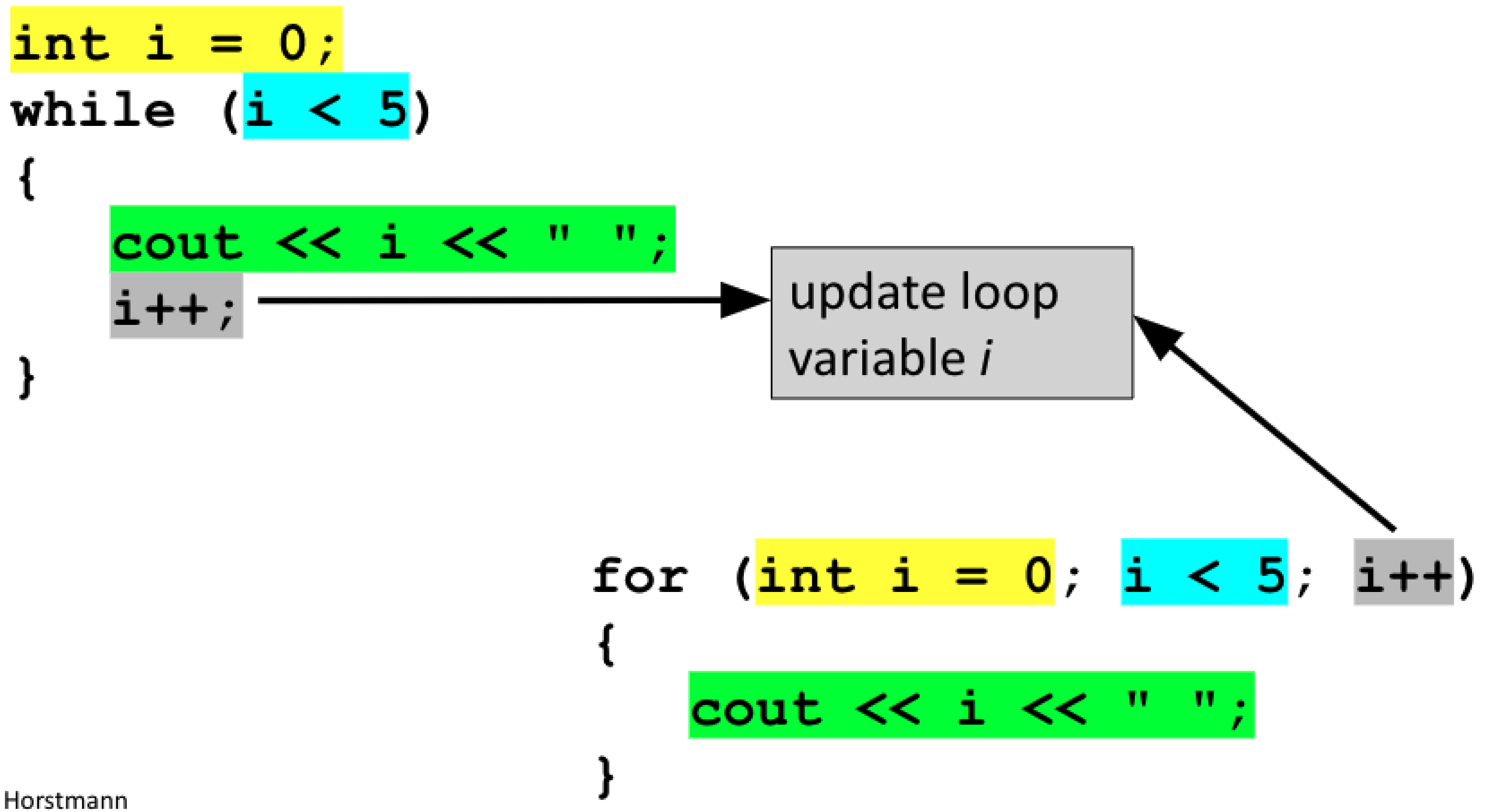


```
int i = 0;
while (i < 5)
{
    cout << i << " ";
    i++;
}
```

```
for (int i = 0; i < 5; i++)
{
    cout << i << " ";
}
```

Brief C++ by Cay Horstmann

Converting from a while loop to a for loop



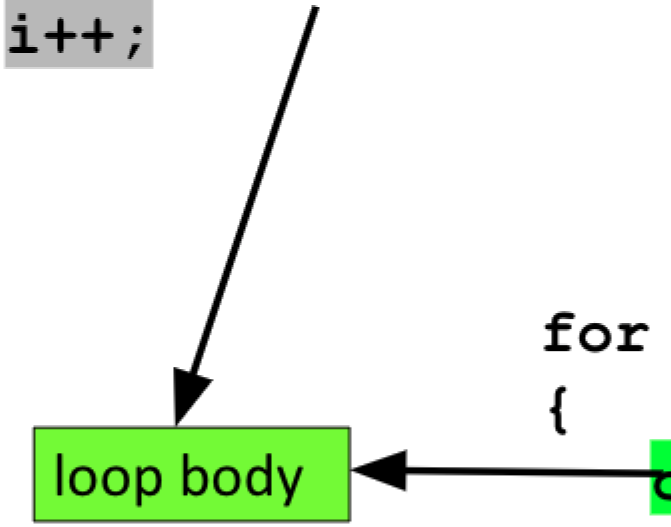
```
int i = 0;
while (i < 5)
{
    cout << i << " ";
    i++;
}
```

```
for (int i = 0; i < 5; i++)
{
    cout << i << " ";
}
```

Brief C++ by Cay Horstmann

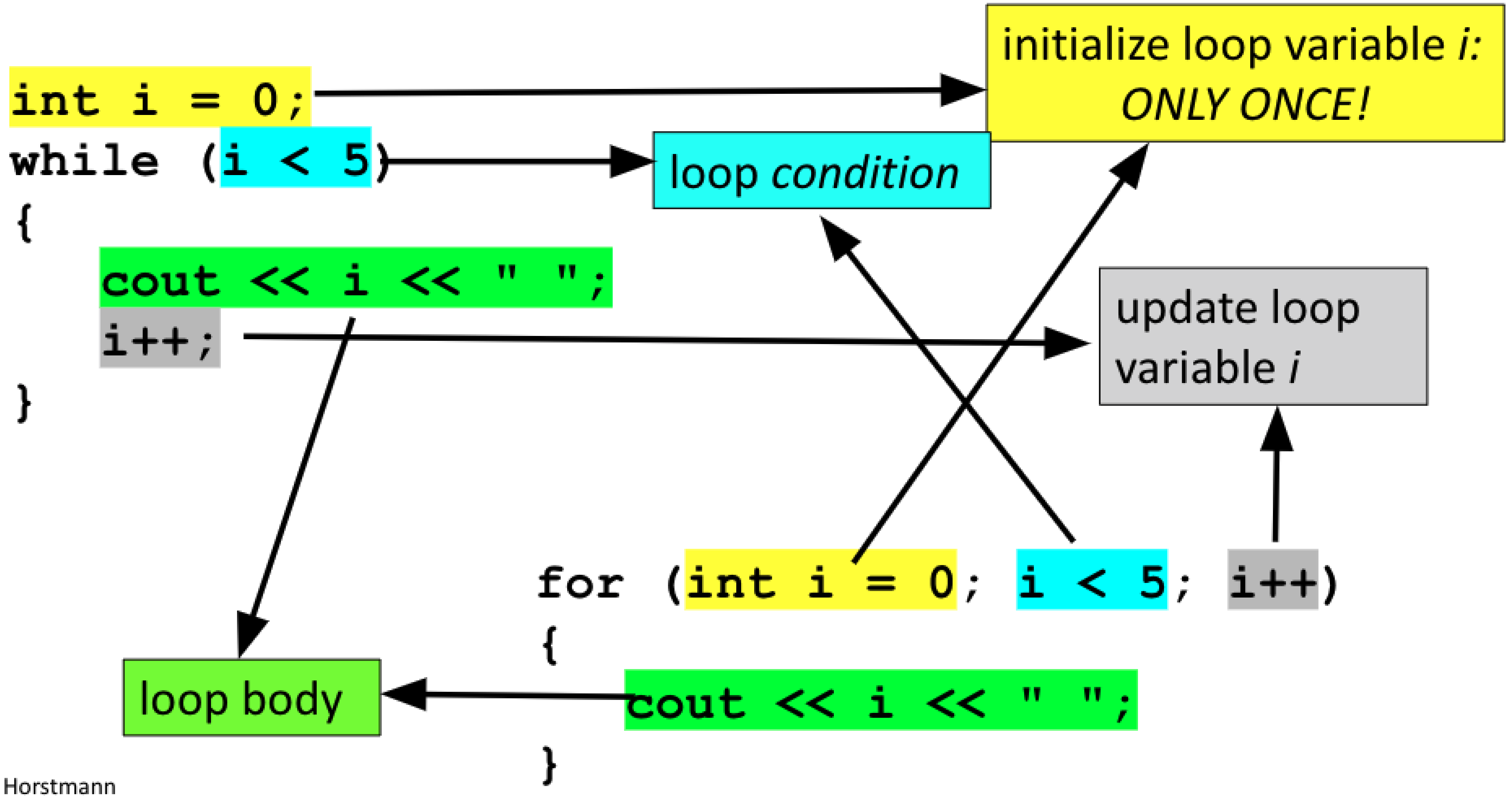
Converting from a while loop to a for loop

```
int i = 0;
while (i < 5)
{
    cout << i << " ";
    i++;
}
```



```
for (int i = 0; i < 5; i++)
{
    cout << i << " ";
}
```

Converting from a while loop to a for loop



```
int i = 0;
while (i < 5)
{
    cout << i << " ";
    i++;
}
```

```
for (int i = 0; i < 5; i++)
{
    cout << i << " ";
}
```

Traversing a string with loops

```
int main()
{
  string str = "ABC";
  for(int i=0; i < str.length();i++)
  {
    cout << str[i] << endl;
  }
  return 0;
}
```

Common Loop Algorithms: Counting Matches:

```
//Counting chars in a string
int spaces = 0;
for (int i = 0; i < str.length();i++)
  {
    if (str.substr(i, 1)== " ")
      {
        spaces++;
      }
  }
```

Common Loop Algorithms: Counting Matches:

```
//Counting words in a user
input sequence
int short_words = 0;
string input;
while (cin >> input)
  {
    if (input.length() <= 3)
      {
        short_words++;
      }
  }
```

Common Loop Algorithms: Finding First Location

```
//Find the location in a string of first space char
bool found = false; //flag=false says "not found yet"
int position = 0;

while (!found && position < str.length())
  {
    string ch = str.substr(position, 1);
    if (ch == " ")
      {
        found = true;
      }
    else
      {
        position++;
      }
  }
```

Common Loop Algorithms: Prompting Until Matched

```
//Repeat prompt until user enters valid value
bool valid = false; //input not valid yet
double input; //declare input var outside loop,
              //so it will persist after loop exit
while (!valid)
{
  cout << "Please enter a positive value < 100: ";
  cin >> input;
  if (0 < input && input < 100)
    { valid = true; }
  else
    { cout << "Invalid input." << endl; }
}
```

Common Loop Algorithms: Min and Max

```
//Save the min and max values of user input list
// This is a merger of the min and max loops from book
double largest, smallest;
double input;
cin >> largest; //get first value to use in loop
smallest = largest; // copy it.
  // If only 1 entry, it is both smallest and the largest
while (cin >> input)
{
  if (input > largest)
    { largest = input; }
  else if (input < smallest)
    { smallest = input; }
}
```

Common Loop Algorithms: Comparing Adjacent Values

```
//Find adjacent duplicates of user input list
// In a later chapter, we'll show how to use arrays to
// find non-adjacent duplicates
double input;
double previous; //to keep track of prior entry
cin >> previous; //first entry becomes first previous
while (cin >> input)
{
  if (input == previous)
  {
    cout << "Duplicate input" << endl;
  }
  previous = input; //save it to compare to next input
}
```

Worked Example 4.1: Loop to Remove Chars from string

// worked_example_1/ccnumber.cpp

// Removes all spaces or dashes from a string

Two options:

1. Create a new string that will have the answer
   - add/concatenate in there only the characters you want to keep
2. Modify the original string variable
   - Keep reassigning new values to the original string by piecing together substrings from before and after any character you don't want

Worked Example 4.1: Loop to Remove Chars from string

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  string credit_card_number = "4123- 5678 - 9012 - 3450";
  int i = 0;
  while (i < credit_card_number.length())
  {
    string ch = credit_card_number.substr(i, 1);
    if (ch == " " || ch == "-") //must remove char
    {
      string before = credit_card_number.substr(0, i);
      string after = credit_card_number.substr(i + 1);
      credit_card_number = before + after;
    }
    else // no need to remove it, go to next char
      { i++; }
  }
  cout << credit_card_number << endl;
  return 0;
}
```