COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 6 - Class 2: Loops Part II

Administrivia

- *Small Programming Assignment 2* and *Large Programming Assignment 1* will come out today!
- All assignments will be returned this week.
- Exams grades will be released by Friday.

Today's Agenda

1. Quick Recap of past concepts
2. More on Loops!

Quick Review

Compound Assignment Operators

- You may have noticed instructions where variable have assignment statement that involves itself.
  - `var1 = var1 + 1;`
  - `var2 = var2 - 2;`
- C, this can be rewritten as a compound statement.
  - `+: +=  e.g., var1 += 1;`
  - `-: -=  e.g., var2 -= 2;`
  - `*: *=`
  - `/: /=`
  - `%: %=`

Examples of Compound Assignment Operators

Compound Assignment Operators	
`count_emp = count_emp + 1;`	`count_emp += 1;`
`time = time - 1;`	`time -= 1;`
`total_time = total_time + times;`	`total_time += times;`
`product = product * item;`	`product *= item;`
`n = n * (x + 1);`	`n *= (x + 1);`

In-Class Exercise Solutions

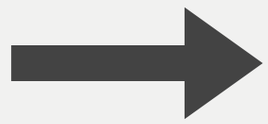
- Write the equivalents for the following statements using compound assignment operators:

```
s = s / 5;

q = q * (n + 4);

z = z - x * y;

t = t + (u % v);
```



```
s /= 5;

q *= (n + 4);

z -= (x * y);

t += (u % v);
```

Operator Precedence

Operator	Precedence
`function calls`	Highest
`! + - & (unary)`	
`* / %`	
`+ -`	
`< <= >= >`	
`!= ==`	
`&&`	
`||`	
`(=)`	Lowest

Operator Precedence

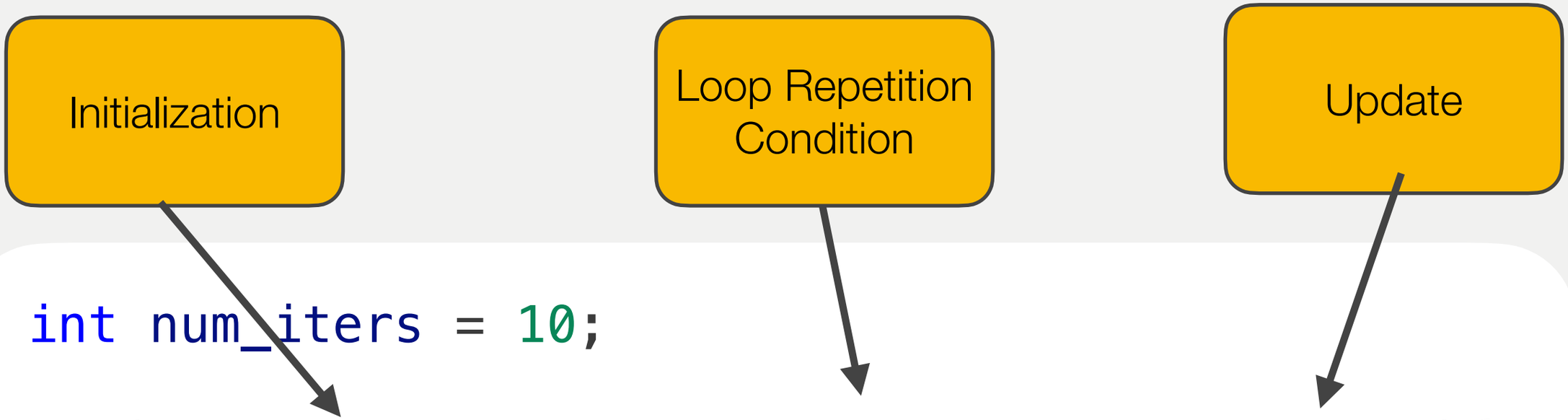
Operator	Precedence
`function calls`	Highest
`! + - & (unary)`	
`* / %`	
`+ -`	
`< <= >= >`	
`!= ==`	
`&&`	
`||`	
`(=, +=, -=, *= …)`	Lowest

More Loops!

The For Statement

- While loops are very useful when programmers aren't sure how many time a set of instructions should be executed.
- For loops are another type of loops where we know exactly how many times a group set of instructions needs to be executed
- There are three components to the for loop:
  - Initialization of the loop control variable
  - Test of the loop repetition condition
  - Update to the loop control variable

The For Statement



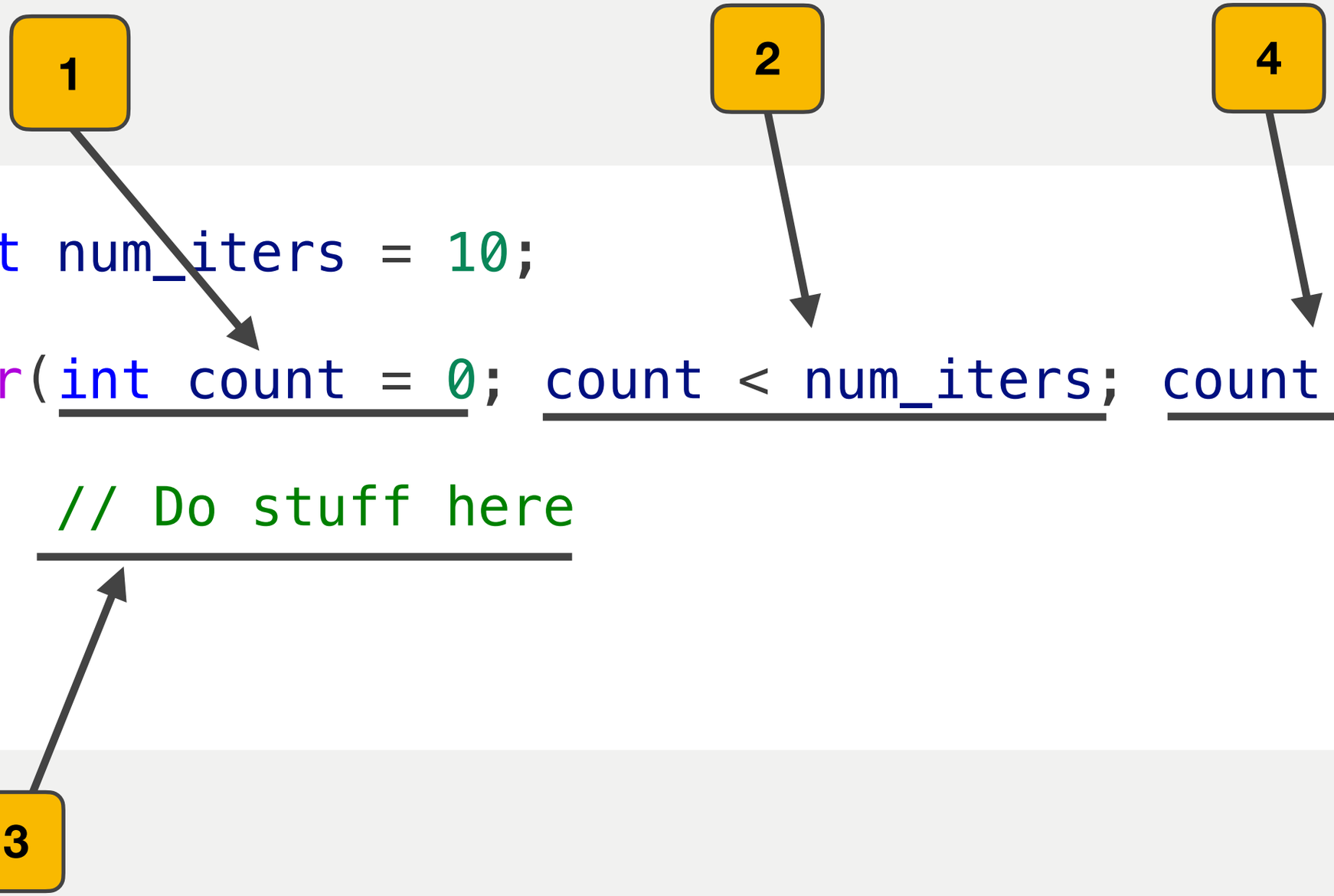

```
int num_iters = 10;

for(int count = 0; count < num_iters; count ++){

    // Do stuff here

}
```

For Loop Control Flow



Increment and Decrement Operators

- C provides an alternative when writing an increment and decrement by 1 statement.

- `counter = counter + 1`; can be rewritten as `counter++`;

- `counter = counter - 1`; can be rewritten as `counter--`;

- Pre increment/ Pre decrement (`--counter`;)

- Post increment/Post decrement (`counter++`)

Infinite Loops

- Loops makes it much easier for programmers for writing repetition.
- However, programmers must be careful with designing loops!
- If loops are not properly setup to terminate at some point, they could be stuck in an infinite loop!!!
- Infinite loop will cause the compiler to execute until RAM Space is filled, causing the program to crash!

```
while(1){
    printf(“AHHHH!!!\n");
}
```

Nested Loops

- The past examples we have only observed one loop. However, it is possible to have loops within loops (nested loops)

- Nested loops have the following terminology:

  - Outer loop

  - Inner loop

```
for(int x = 0; x < 5; ++x){ // Outer Loop

    for(int y = 0; y < 2; ++y){ // Inner Loop

        printf("x = %d\n", x);
        printf("y = %d\n", y);
    }


}
```

Do-While Loops

- For loops allow programmers to execute instructions a set number times.
- While loops allow programmers to execute instructions multiple times until a condition is met.
- Do-while loops allow programmer to execute instructions multiple times until a condition is met, however the instructions will be executed once at least.

Do-While Loop Example

1

```
char letter_choice;

do{
printf("Enter a latter from A through E: ");
scanf(" %c", &letter_choice);

}while(letter_choice >= 'A' && letter_choice <= 'E');
```

2

Acknowledgements

Slides adapted from Dr. Andrew Steinberg's COP 3223H course