CPS 188

Computer Programming Fundamentals

Prof. Alex Ufkes

Topic 5.1: Basic looping

Toronto Metropolitan University

Notice!

Obligatory copyright notice in the age of digital delivery and online classrooms:

The copyright to this original work is held by Alex Ufkes. Students registered in course CPS 188 can use this material for the purposes of this course but no other use is permitted, and there can be no sale or transfer or use of the work for any other purpose without explicit permission of Alex Ufkes.

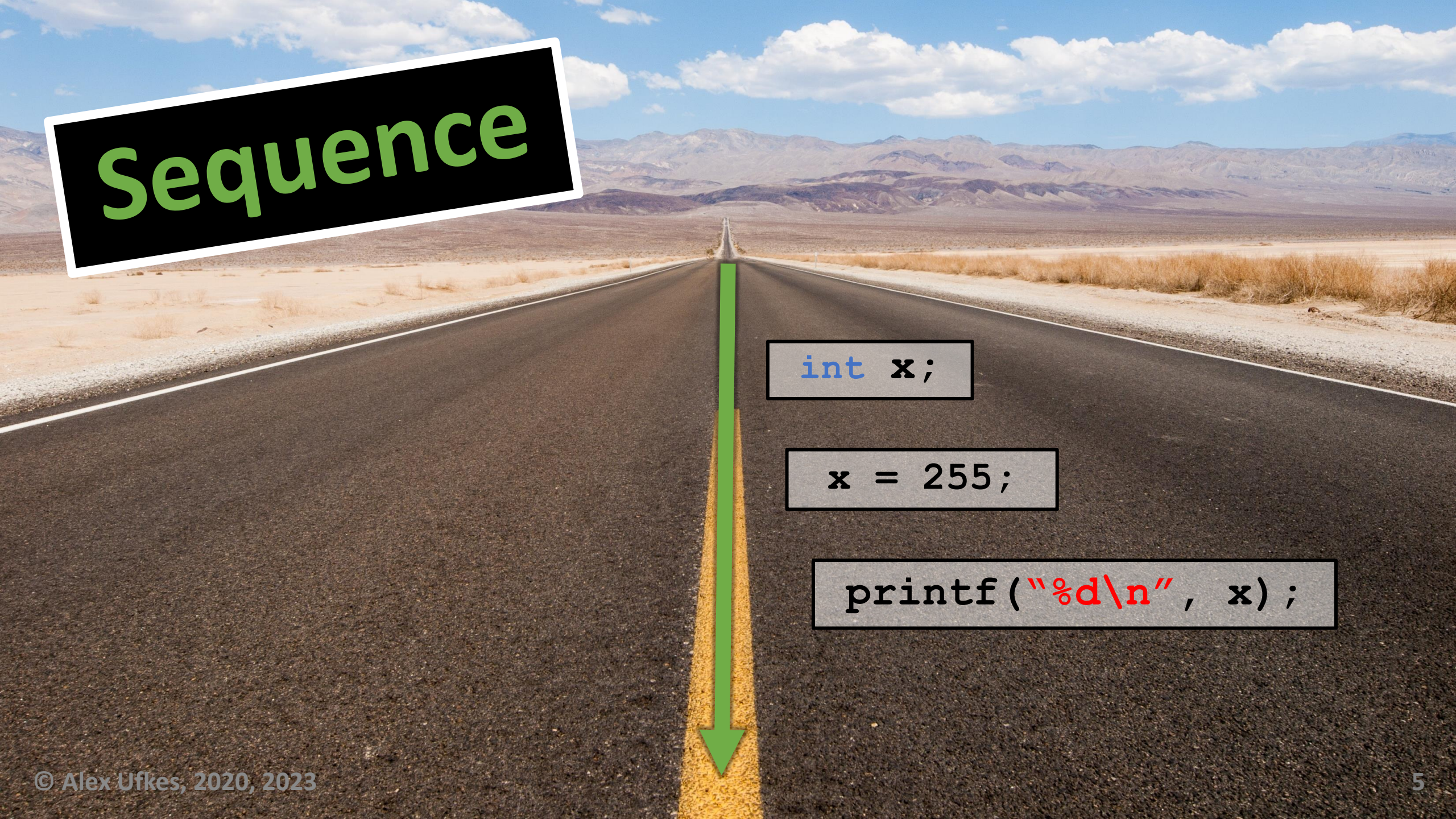
Today

Looping #1

Iteration basics, `for` loops,
`while` loops, `do/while` loops

Control Structures

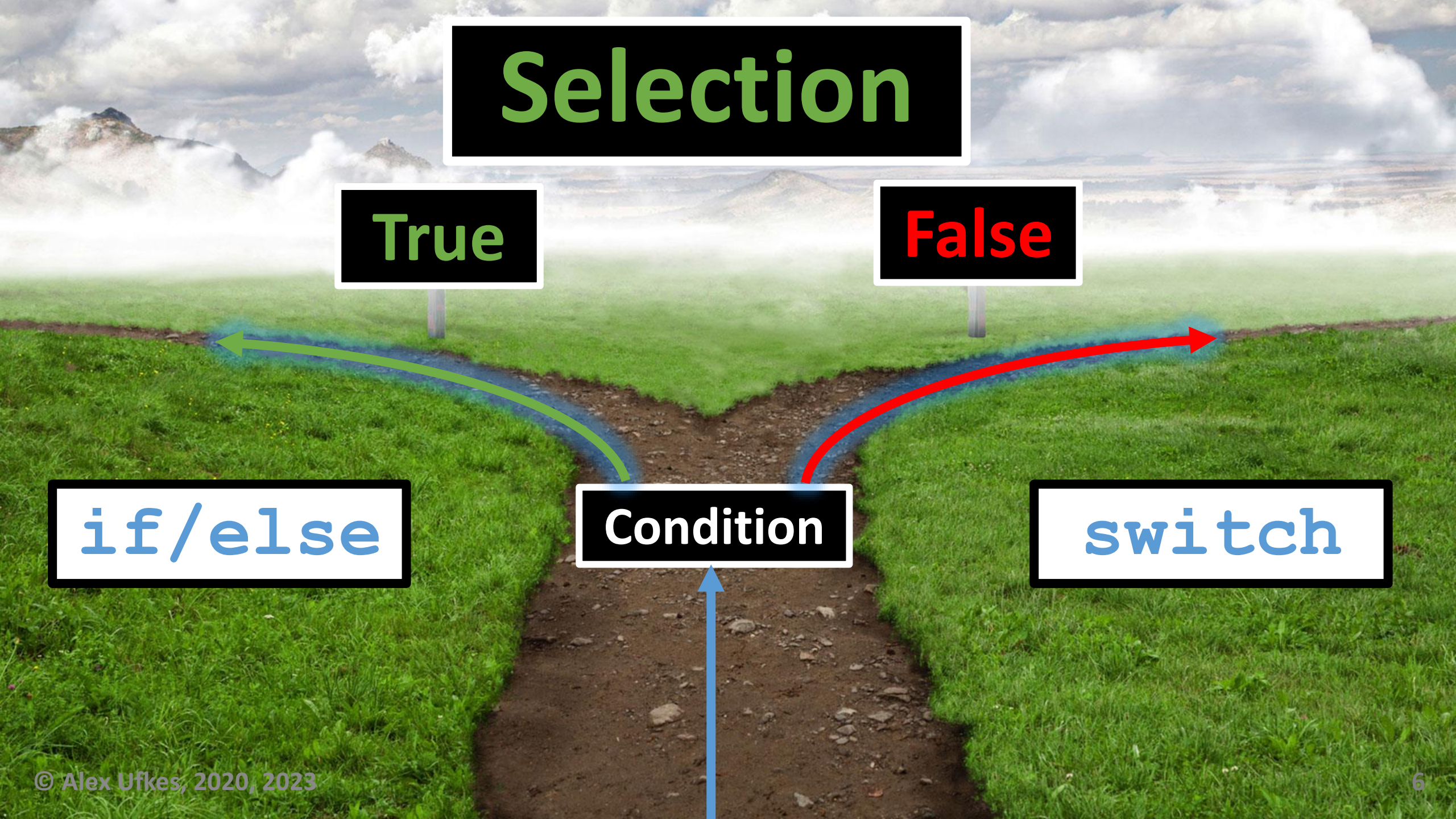
Sequence



```
int x;
```

```
x = 255;
```

```
printf("%d\n", x);
```



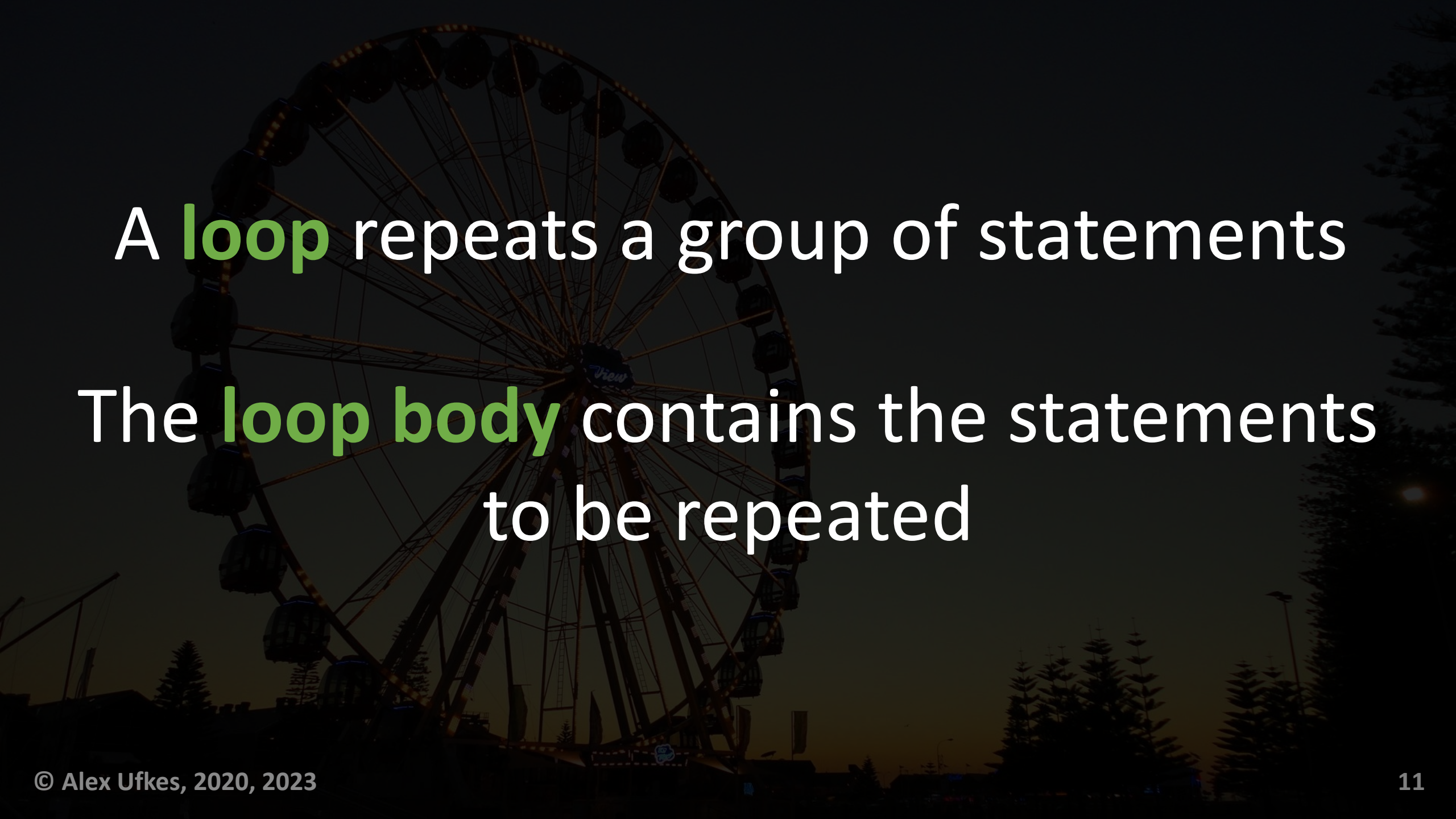
Write a code snippet to print “Hello, World!” five times to the screen.

```
printf(“Hello, World!\n“);
printf(“Hello, World!\n“);
printf(“Hello, World!\n“);
printf(“Hello, World!\n“);
printf(“Hello, World!\n“);
```

Write a code snippet to print "Hello, World!" <u>fifty</u> times to the screen.

```
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
printf("Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n Hello, World!\n");
```

Write a code snippet to print “Hello, World!” FIVE THOUSAND times to the screen.



A **loop** repeats a group of statements

The **loop body** contains the statements to be repeated

3

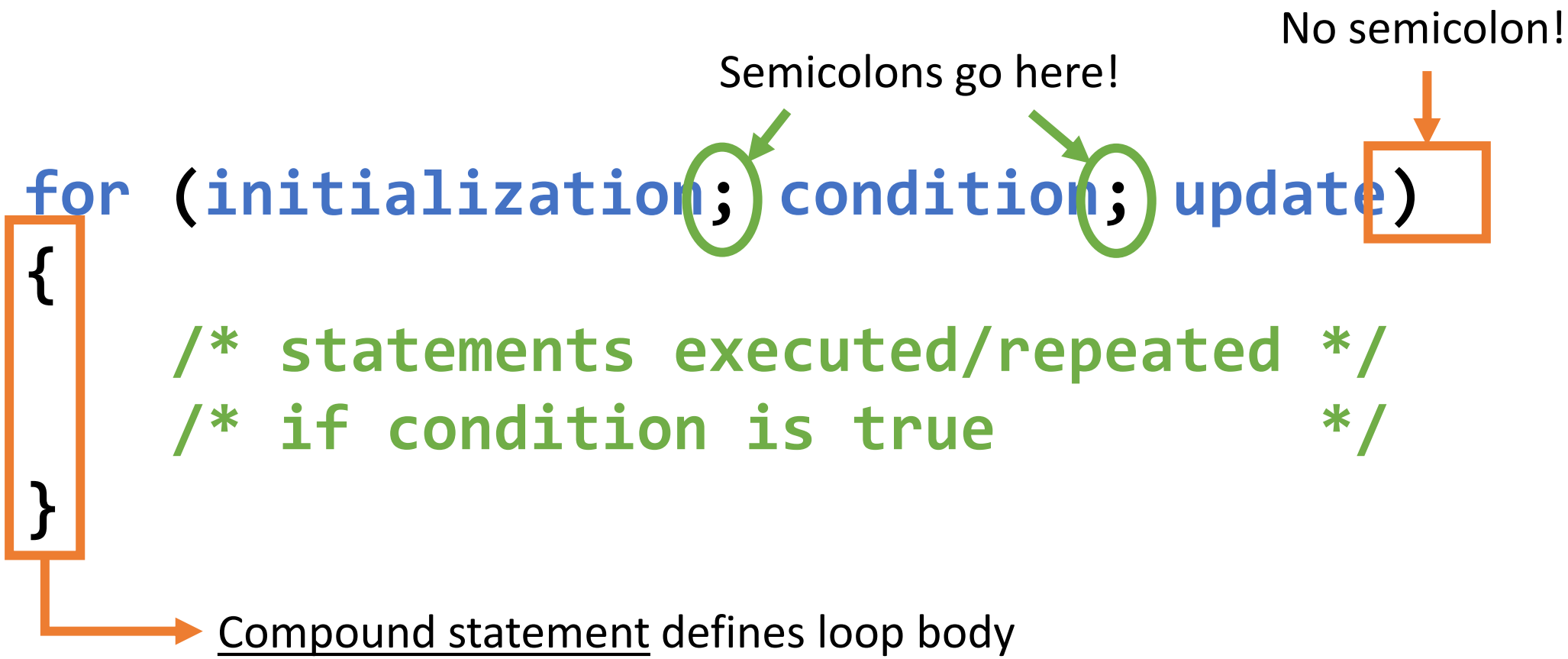
Loop Structures

1) `for` loop

2) `while` loop

3) `do/while` loop

`for` loop



```
int i;

for (i = 1; i <= 3; i++)
{
    printf("Hello, World!\n");
}
```

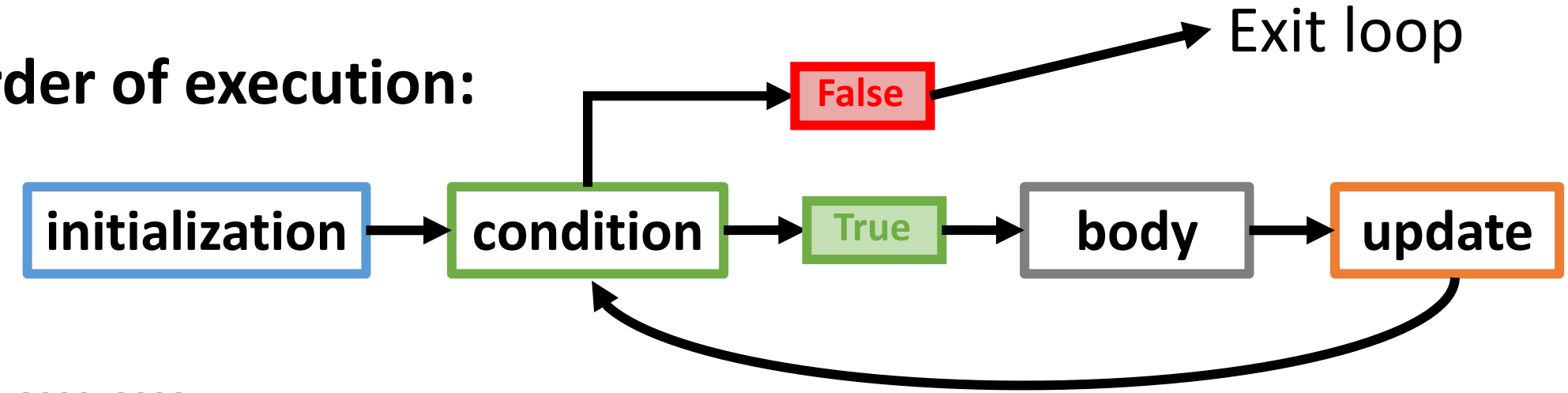
initialization

condition

update

Loop body

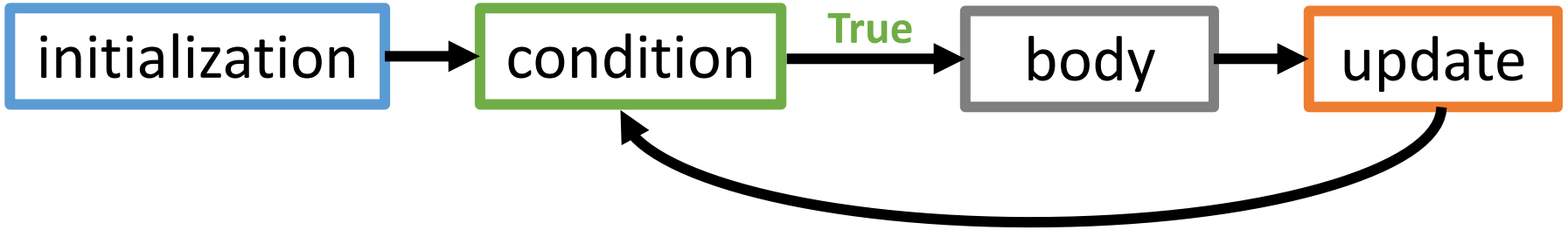
Order of execution:



```
int i;

for (i = 1; i <= 3; i++)
{
    printf("Hello, World!\n");
}
```

Output:

```
Hello, World!
Hello, World!
Hello, World!
```




```
int val;

for (val = 1; val <= 2; val++)
    printf("inside loop: val = %d\n", val);

printf("outside loop: val = %d\n", val);
```

Output:

```
inside loop: val = 1
inside loop: val = 2
outside loop: val = 3
```

What is the output?

```
int n;
for (n = 50; n <= 100; n++)
{
    printf("%d\n", n);
}
```

Numbers between 50-100, inclusive

Write code snippet using a for loop to do the following:

Print all even numbers between 0 and 100.

```
int n;
for (n = 0; n <= 100; n += 2)
{
    printf("%d\n", n);
}
```

Increase **n** by **2**

Another (less efficient) option:

```
int n;
for (n = 0; n <= 100; n++)
{
    if (n % 2 == 0)     /* if(!(n % 2)) */
    {
        printf("%d\n", n);
    }
}
```

Control structures can be nested inside each other any way you like

Write code snippet using a for loop to do the following:

Print all numbers between 0 and 100 in *descending* order.

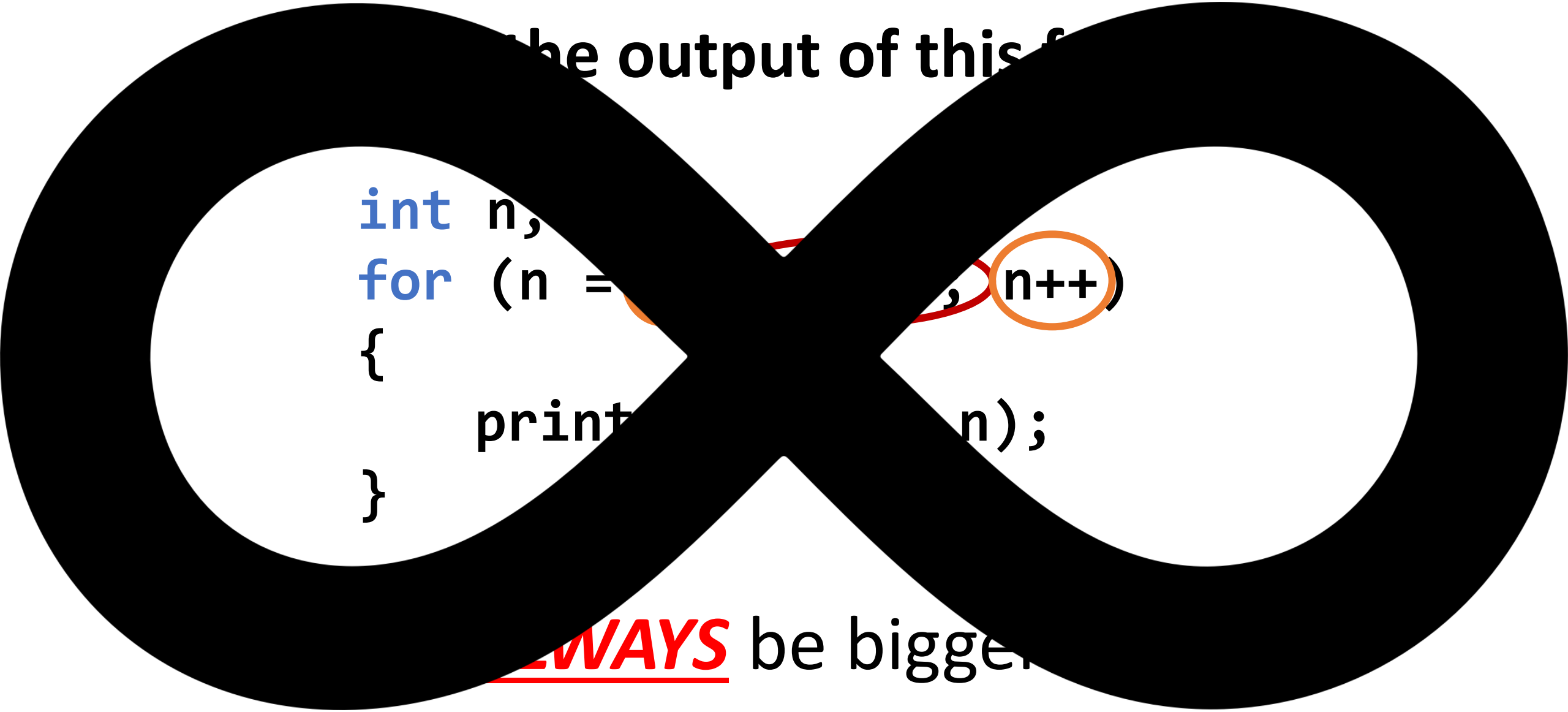
```
int n;
for (n = 100; n >= 0; n--)
{
    printf("%d\n", n);
}
```

Decrement!

What is the output of this for loop?

```
int n;
for (n = 1; n >= 0; n++)
{
    printf("%d\n", n);
}
```

n will ***ALWAYS*** be bigger than 0!



he output of this f

```
int n,
for (n =                    , n++)
{
    print                   n);
}
```

_EWAYS be bigge

Infinite Loops

```
int n;
for (n = 1; n >= 0; n++) {
    printf("%d\n", n);
}
```

This loop is not *truly* infinite. Eventually, `n` will overflow and become negative, thus making the condition `n >= 0` false.

Infinite Loops

Let's cause an overflow in a short amount of time:

```
int n;
for (n = 1; n >= 0; n += 100000) {
    printf("%d\n", n);
}
printf("%d\n", n);
```

Infinite Loops

Can we make a loop that doesn't end due to overflow?

```
int n;
for (n = 1; 1; n++) {
    printf("%d\n", n);
}
```

Condition is always true. Variable `n` will still overflow, but it won't cause the loop to exit.

Beware Infinite Loops!

Quincy 2005 - [infiniteLoop]

File Edit View Project Debug Tools Window Help

```
#include <stdio.h>

int main (void)
{
    int n;

    for(n = 1; n >= 0; n++)
    {
        printf("%d\n", n);
    }

}
```

Press F1 for help

How about something more complicated?

Write code snippet using a for loop to do the following:

Calculate the factorial of a number entered by the user.

Factorial:

The product of all positive integers less than or equal to a given non-negative number.

$$5! = 5*4*3*2*1$$

Write code snippet using a for loop to do the following:

Calculate the factorial of a number entered by the user.

Where to start?

This is doable!

```
int num;
printf("Enter a number: ");
scanf("%d", &num);
```

Yay!

Write code snippet using a for loop to do the following:

Calculate the factorial of a number entered by the user.

Now what? We know we need a for loop.

```
int num, i, fact = 1;
printf("Enter a number: ");
scanf("%d", &num);

for (                    )
{

}
```

And how about some extra variables?

One to control the loop, and another to store the result.

Write code snippet using a for loop to do the following:

Calculate the factorial of a number entered by the user.

```
int num, i, fact = 1;
printf("Enter a number: ");
scanf("%d", &num);

for ( i = 1;  i <= num; i++)
{
    fact = fact * i;
}
```

Multiply **fact** by **i** at each iteration.

To calculate the factorial of **num**, we need to multiply together every number from 1 to **num**.

Initialize: `i = 1`

Condition: `i <= num`

Update: `i++`

Write code snippet using a for loop to do the following:

Calculate the factorial of a number entered by the user.

```
int num, i, fact = 1;
printf("Enter a number: ");
scanf("%d", &num);

for (i = 1; i <= num; i++)
{
    fact = fact * i;
}

printf("Factorial of %d is %d \n", num, fact);
```

Finally,

Print the result.

```
int num = 5, i, fact = 1;
for (i = 1; i <= num; i++) {
      fact = fact * i;
      printf("After iteration %d, fact = %d\n", i, fact);
}
printf("\nFactorial of %d is %d \n", num, fact);
```

Let's simplify for the sake of demonstration. Let num = 5.

Output:

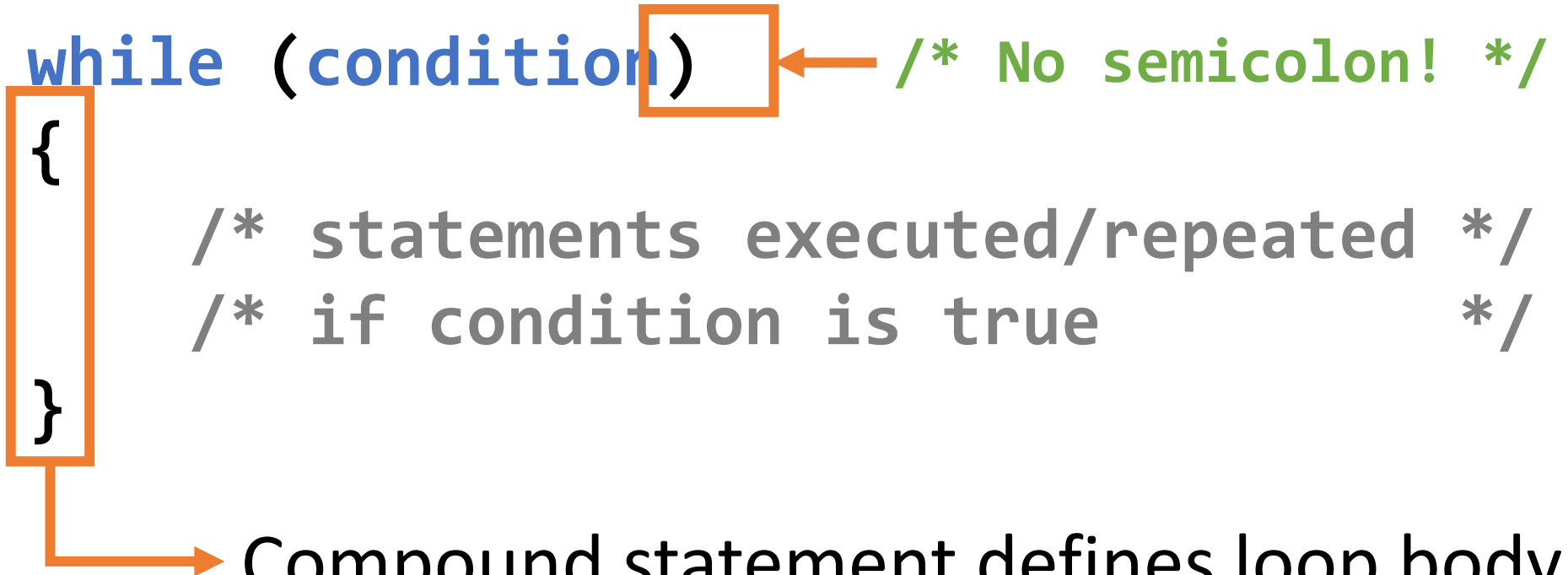
```
After iteration 1, fact = 1      /* fact =  1*1 = 1   */
After iteration 2, fact = 2      /* fact =  1*2 = 2   */
After iteration 3, fact = 6      /* fact =  2*3 = 6   */
After iteration 4, fact = 24     /* fact =  6*4 = 24  */
After iteration 5, fact = 120    /* fact = 24*5 = 120 */

Factorial of 5 is 120
```

1) `for` loop

2) `while` loop

3) `do/while` loop

`while` loop



```c
int n = 1;              /* initialize */
while (n <= 2)          /* condition  */
{
    printf("%d\n", n);
    n = n + 1;          /* update     */
}
```

Unlike a `for` loop, a `while` loop ***only*** requires that we provide a condition for it to compile.

We must be very careful to include the initialization and the update steps on our own.

```
int n = 1, a = 1;                /* initialize */
while (n <= 2)                   /* condition  */
{
    a = a + 1;                   /* update…?   */
    printf("%d\n", a);
    printf("%d\n", a*2);
    n += 1;
}
```

Is there something wrong with this loop?

`n` is never updated!

Without an update statement, we have an infinite loop!

Write code snippet using a while loop to do the following:

Print all odd numbers between 0 and 100.

```
int n = 1;              /* Initialize */

while ( n <= 100)       /* Condition  */
{
    printf("%d\n", n);
    n = n + 2;          /* Update     */
}
```

Write code snippet using a while loop to do the following:

Print all odd numbers between 1 and 30 divisible by 3 or 5.

```
int n = 1;  /* Initialize */
while ( n <= 30) {  /* Condition */
    if (n%3 == 0 || n%5 == 0)
        printf("%d ", n);
    n+=2;/* Update */
}
```

Console:

```
3 5 9 15 21 25 27
```

Write code snippet using a while loop to do the following:

Print all perfect squares less than 100.

```
int n = 1;                  /* Initialize */
while ( n*n < 100 )         /* Condition  */
{
    printf("%d ", n*n);
    n++;                    /* Update     */
}
```

Console:

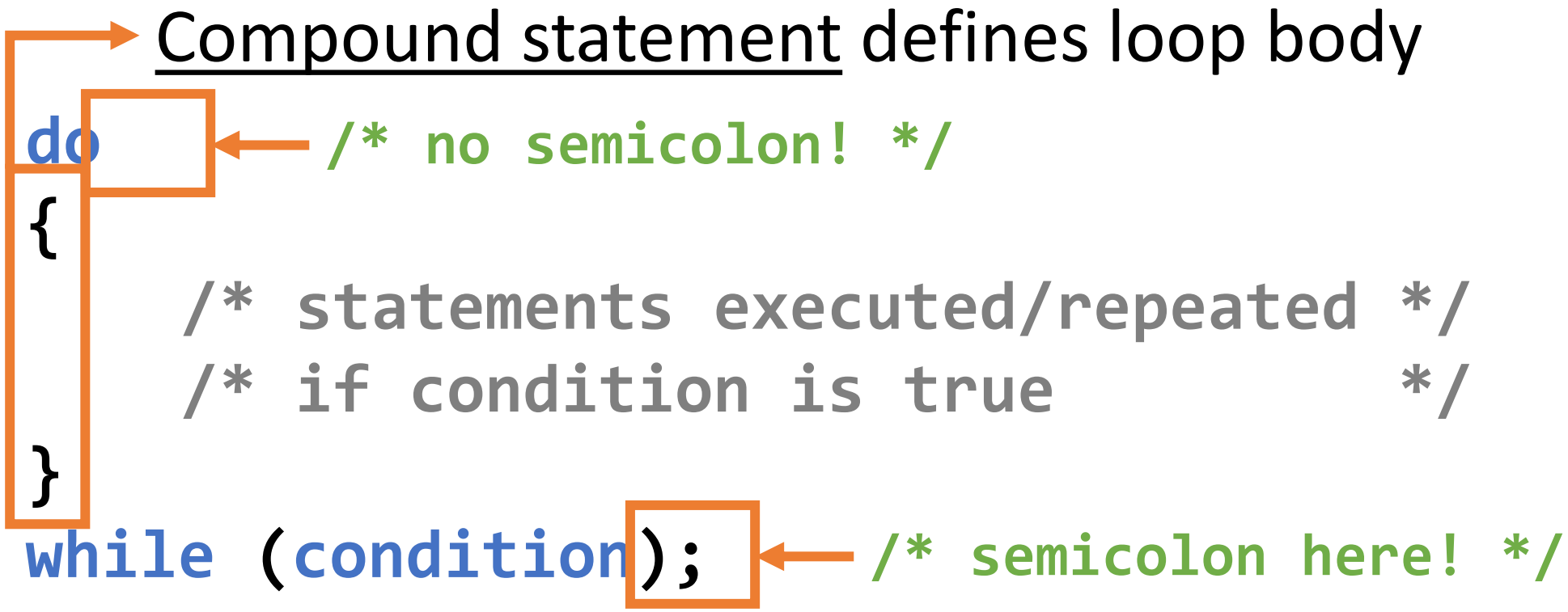
```
1 4 9 16 25 36 49 64 81
```

1) `for` loop

2) `while` loop

3) `do/while` loop

`do while` loop



```
int n = 1;              /* initialize */

do
{
    printf("%d ", n);
    n = n + 1;          /* update     */
}
while (n <= 5);         /* condition  */
```

Output:

```
1 2 3 4 5
```

`while` VS `do/while`

```
while (condition) {
  /* statements */
}
```

Condition checked at the ***start***.
If condition is initially **false**, body never executes.

Condition checked at the ***end***.
If condition is initially **false**, body will still execute **once**.

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

The body of a `do/while` loop will execute **at least once**.

```c
int n = 5;

while (n <= 2)
{
  printf("%d\n", n);
  n = n + 1;
}
printf("Done loop.\n");
```

Output:

```
Done loop.
```

```c
int n = 5;

do {
  printf("%d\n", n);
  n = n + 1;
}
while (n <= 2);
printf("Done loop.\n");
```

Output:

```
5
Done loop.
```

When is it useful to execute at least once?

Two words:

Input Validation

Write code snippet to do the following:

Read in numbers until a negative value is entered.

```
int value;

do {
    printf("Enter an integer: ");
    scanf("%d", &value);
}
while ( value >= 0 );
```

`for` loop

Use when the number of iterations is known in advance.

```
int i;
for (i = 1; i <= 5; i++) {
    printf("Hello, World!\n");
}
```

A `for` loop conveniently integrates the initialization, condition, and update steps into a single line.

`do/while` loop

Use when the loop body must execute at least once.

```
int value;
do {
    printf("Enter an integer: ");
    scanf("%d", &value);
} while (value >=0);
```

A `do/while` loop is ideal for input validation.

`while` loop

Use in all other situations.

```
int n = 1;
while (n <= 2) {
    printf("%d\n", n);
    n = n + 1;
}
```

A `while` loop can be adapted to any looping task.

A Loop is as Loop is a Loop

If the task can be solved with a `while` loop, it can also be solved with a `for` loop, and so on.

By adjusting initializations, conditions, and updates, every loop can be tweaked to be the functional equivalent of another.

This is FANTASTIC practice! Try solving a looping problem with each of three loop styles.

`break`

vs.

`continue`

Remember break?

```
int n = 1;
while (1) {          /* Always true! */
    printf("%d ", n++);
    if (n > 5)
        break;       /* break exits the loop */
}
```

Console:

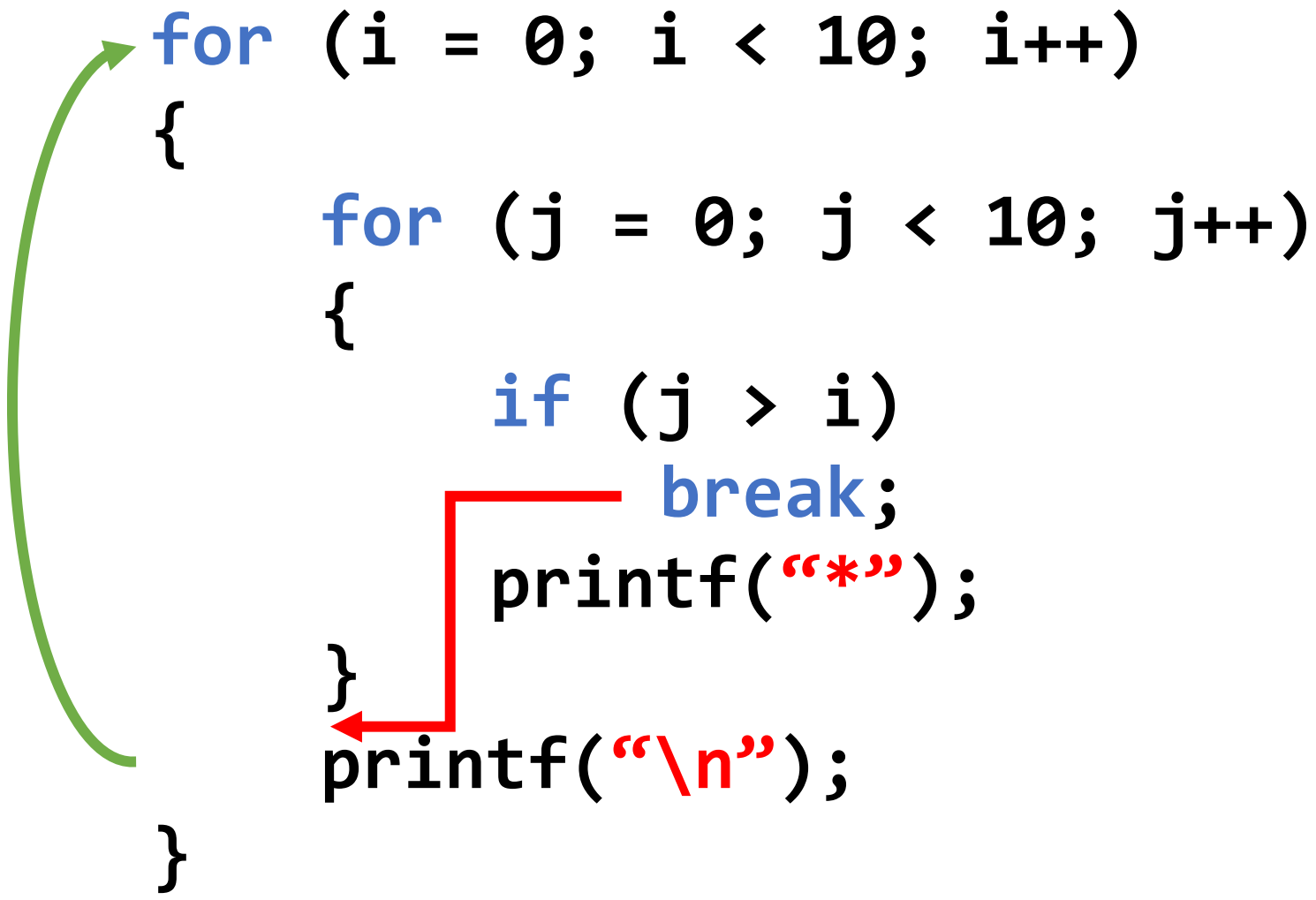
```
1 2 3 4 5
```

A **break** statement jumps out of the *innermost* enclosing loop of the statement

```
int i = 1;
while (1)  /* Condition is always true */
{
     if (i >= 6)    }  /* if + break is used
          break;    }     to exit the loop  */
     printf("%d ", i);
     i = i + 1;
}
```

Output: 1 2 3 4 5

A **break** statement jumps out of the *innermost* enclosing loop of the statement



`break` VS `continue`

The `continue` statement ends the *current iteration* of a loop

Contrast with the `break` statement, which exits the loop entirely.

The `continue` statement skips the *current iteration* of a loop

```
for (n = 1; n <= 10; n++)
{
    if (n % 2 != 0)
        continue;
    printf("%d ", n);
}
```

Continue if n is odd

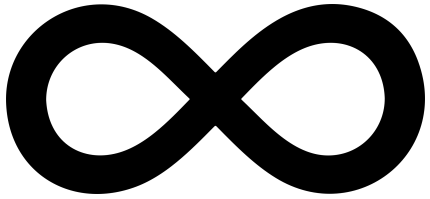
Output: `2 4 6 8 10`

Spot the ERROR?

```c
#include <stdio.h>

int main(void)
{
    int n, x = -3;
    for (n = x, n <= 0, n += 1)
        printf("%d\n", n);

    return 0;
}
```

Should be semicolons, not commas

```c
#include <stdio.h>

int main(void)
{
    int n;
    for (n = 1; n != 50; n *= 2) {
        printf("%d\n", n);
    }

    return 0;
}
```



```c
#include <stdio.h>

int main (void)
{
    int x = 5;
    while( x > 0 );
        x--;

    return 55;
}
```

```
#include <stdio.h>

int main(void)
{
    for (i = 1; i <= 12; i++)
    {
        printf("%d\n",i);
    }

    return (0);
}
```

Variable not declared!

```c
#include <stdio.h>

int main(void)
{
    int n;

    do
    {
        printf("Enter a number ");
        printf("between 1 and 9: ");
        scanf ("%d", &n);
    }while (n < 1 || n > 9)

  return (0);
}
```

Questions?