C++ Day 4 Commands, Functions, and Tricks:

---

const:

- Purpose: declares a variable as a constant, unchanging value.
- Examples:
  - create a constant called pi and set the value to 3.1415926

```
const double pi=3.1415926;
```

Arithmetic operators:

- Purpose: perform basic arithmetic operations.
- Note: only the math operations shown below are included in baseline C++. To add many other relatively common math functions (such as sine, absolute value, and exponentiation), you must `#include <cmath>`. A list of `cmath` functions: [www.cplusplus.com/reference/cmath/].
- Examples:
  - Add, subtract, multiply, and then divide "num_of_chickens" by "num_of_humans"

```
num_of_chickens + num_of_humans;
num_of_chickens - num_of_humans;
num_of_chickens * num_of_humans;
num_of_chickens / num_of_humans;
```

  - Get the remainder of "num_of_chickens" divided by "num_of_humans"

```
num_of_chickens % num_of_humans;
```

---

increment operator (++) & decrement operator (--):

- Purpose: increase (++) or decrease (--) by 1 the value of a numeric variable (almost always one of the integer types)
- Note: the operator can come before or after the variable name. If it comes before, then the change in value happens *before* the rest of the current command. If it comes after the variable name, then the change happens *after* the rest of the current command
- Examples:
  - set the value of "num_old_chickens" to "num_eggs_hatched" and *then* increase the value of "num_eggs_hatched" by 1

```
num_old_chickens = num_eggs_hatched++;
```

  - decrease the value of "num_eggs" by one and then set the value of "num_omlettes_tomorrow" to the ***new*** value of "num_eggs" divided by 2

```
num_omlettes_tomorrow = --num_eggs / 2;
```

---

rand():

- Purpose: get a “random” positive integer between 0 and some large maximum value.
- Note: the maximum value depends on the development environment. In Dev-C++, it’s 32,767. The value is stored in the constant "`RAND_MAX`"
- Note: this function is included in header file `<cstdlib>`
- Examples:
  - print to the screen a random integer between 0 and 100

```
#include <cstdlib>

...

cout << rand() % 101 << endl;
```

  - print to the screen a random floating point number between 0 and 10

```
#include <cstdlib>

...

cout << 10.0 * rand() / RAND_MAX << endl;
```

  - create the variable "dist_closest_chicken" and set its value to a random integer between -10 and 10

```
#include <cstdlib>

...

int dist_closest_chicken = ((rand() % 21) - 10);
```

---

srand():

- Purpose: seed the random number generator.
- Note: without using a different seed value, the `rand()` function will return the same "random" numbers each time a program is run.
- Note: a seed value of the current time is commonly used, since that will be different on each run. The current time can be retrieved using the `time(0)` function, which is found in the `<ctime>` library
- Examples:
  - output to the screen a "truly" random integer between 1 and 6

```
#include <cstdlib>
#include <ctime>

...

srand(time(0));

...

cout << (rand() % 6 + 1) << endl;
```

Boolean variables:

- Purpose: Boolean variables are simply a different type of variable. They can store only two values: true or false.
- Examples:
  - create a Boolean variable named "chickens_are_angry" and initialize it to true

```
bool chickens_are_angry = true;
```

Logical combinators (&&, ||):

- Purpose: used to combine multiple true or false values using a logical "and" (&&) or "or" (||)
- Note: Parentheses are used below to clarify the statements. They are recommended but not strictly necessary.
- Examples:
  - evaluate to "true" if "num_chickens" is greater than 10 AND the bool variable named "chickens_are_angry" has value true

```
(num_chickens > 10) && (chickens_are_angry);
```

  - evaluate to "true" if "num_humans" is greater than "num_chickens" OR if "num_humans" minus "num_chickens" is less than or equal to 2 * "num_weapons"

```
(num_humans > num_chickens) || (num_humans - num_chickens <= 2 *
num_weapons);
```

---

exclamation point (!):

- Purpose: a logical "not" statement, i.e., it converts a value of true to false, and vice-versa
- Examples:
  - store in bool variable "is_there_hope" the value "false" if "chance_of_escape" is less than 0.5 AND the bool variable "do_they_see_me" has value true; otherwise store "true":

```
is_there_hope = ! ((chance_of_escape < 0.5) && (do_they_see_me));
```

Various relationship operators (<, <=, >, >=, ==, !=):

- Purpose: compare two objects and output true or false depending on the result
- Examples:
  - Compare in a few different ways the values stored in two variables "num_humans" and "num_chickens":

```
num_humans < num_chickens; // true if num_humans is less than num_chickens
num_humans <= num_chickens; // true if num_humans is less than or equal to num_chickens
num_humans > num_chickens; // true if num_humans is more than num_chickens
num_humans >= num_chickens; // true if num_humans is more than or equal to num_chickens
num_humans == num_chickens; // true if num_humans is equal to num_chickens
num_humans != num_chickens; // true if num_humans is not equal to num_chickens
```

---

if & if/else:

- Purpose: branches a program along separate paths depending on whether a statement evaluates as true or false
- Examples:
  - perform a block of code only if variable "size_of_beak" is greater than or equal to 1.5

```
if (size_of_beak >= 1.5)
{
   ...
   [commands to run if true]
   ...
}
```

  - perform one block of code if the variable "size_of_beak" is greater than or equal to 1.5, and a different block of code if it is not

```
if (size_of_beak >= 1.5)
{
   ...
   [commands to run if true]
   ...
}
else
{
   ...
   [commands to run if false]
   ...
}
```