Riddles

- What has a foot but no legs?
- What comes down but never goes up?
- I’m tall when I’m young and I’m short when I’m old. What am I?
- What word becomes shorter when you add two letters to it?
- What occurs once in a minute, twice in a moment and never in one thousand years?
- If I have it, I don’t share it. If I share it, I don’t have it. What is it?

Operators
Type Casting
Input / Output

Luke Sathrum – CSCI 20

Today's Class

- Increment Operators
- Decrement Operators
- Type Casting
- Type Coercion
- Output – **`std::cout`**
- The "magic" formula
- New lines
- Output – **`std::cerr`**
- Input – **`std::cin`**

Increment & Decrement Operators

Increment & Decrement Operator

- Increment Operator

  ++

  - Adds 1 to the value of a variable
  - Can put either before or after the identifier

- Decrement Operator

  - -

  - Subtracts 1 from the value of a variable
  - Can put either before or after the identifier

Operator Example

- `int a = 1, b = 7;`
- `a++;`
- `--b;`
- What is the value of **`a`**?
- What is the value of **`b`**?

Increment in Expressions

- We can use these operators in expressions
  - `2 * (a++)`
  - This returns the value and then increments the variable
- Slightly different if we put the operator first
  - `2 * (++a)`
  - This will increment first and then return the value
- The same goes for the decrement operator

```c
int a = 2;
int value_produced = 2 * (++a);
```

```c
int a = 2;
int value_produced = 2 * (a++);
```

Operator Restrictions

- We can only apply these operators to single variables
- We can't do the following
  - `(x + y)++`
  - `--(x + y)`
  - `++5`

Operator Pitfall

- The order of sub-expressions is NOT guaranteed
- Example
  - **`a = 2;`**
  - **`a + (++a);`**
  - Is the result
    - 2 + 3 = 5?
    - 3 + 3 = 6?

Summary

- The increment operator adds 1 to a variable
- The decrement operator subtracts 1 from a variable
- You can put these operators before or after the variable
- If used in an expression placement of the operator matters

Sample Code

- Increment and Decrement Operators
  - `increment_decrement.cpp`

Type Casting

Type Casting

- **Type Casting** is a way to change a value from one type to another
- Example
  - `int x = 9, y = 2;`
  - `x / y`
    - Gives us the value **4**
  - `x / static_cast<double>(y)`
    - Gives us the value 4.5
- `static_cast<double>(y)` evaluates to `2.0`
- The value of `y` does not change

Type Casting

- `int` → `double`
  - Adds a .0
- `double` → `int`
  - Truncates to whole number
  - Does NOT round
- Four kinds
  - `static_cast<type>(`*Expression*`)`
  - `const_cast<type>(`*Expression*`)`
  - `dynamic_cast<type>(`*Expression*`)`
  - `reinterpret_cast<type>(`*Expression*`)`

Type Coercion

- C++ will sometimes automatically type cast for you
  - Called **Type Coercion**
- Example
  - `double d = 5;`
  - `5` is automatically converted to `5.0`

Type Casting Summary

- Type Casting allows us to change the type of a value
  - Does not permanently change the variable type
- Automatic Type Casting is called Type Coercion

Sample Code

- Type Casting and Type Coercion
  - `type_cast.cpp`

Basic Console Output

Output – `std::cout`

- Use to output text to the console screen
- You may output
  - Strings
  - Variables
  - Expressions
  - Combination of all 3
- We use `<<` to separate each type of output
  - Called the **insertion operator**
  - No space between the two symbols

Output – `std::cout`

```
std::cout << "Hello Reader\n";
std::cout << number_of_games
          << " games played";
```

- We may include arithmetic expressions

```
std::cout << "The total cost is $"
          << (price + tax);
```

Output – `std::cout` – Spaces and Line Breaks

- C++ does not enter any spaces or line breaks for you
- We must manually add spaces to get output to look correct
- We also use `"\n"` or `std::endl` to add line breaks

```
std::cout << "The value is "
          << value << std::endl;
```

Output – `"\n"` vs. `std::endl`

- We use `"\n"` if we are already in quotes
- Otherwise we use `std::endl`
- Example

```
std::cout << "Fuel efficiency is "
             << mpg << " mpg.\n";
std::cout << "Fuel efficiency is "
             << mpg << std::endl;
```

Output - Includes

- In order to use **`std::cout`** we need the following include

```
#include <iostream>
```

- In order to not type the **`std::`** part every time

```
using std::cout;
using std::endl;
```

Output - Formatting

- Doubles may not be in the format you want them to be in
- Example

```
double price = 78.5;
cout << "The price is $" << price << endl;
```

  - What is **`price`** going to output as?

Output - Formatting

- To get our output formatted correctly we use the following

```
cout.setf(std::ios::fixed|std::ios::showpoint);
cout.precision(2);
```

- To change the precision again we just use the last line
  - `cout.precision(4);`

Output – `std::cerr`

- **`std::cerr`** is used the same way as **`std::cout`**
- Sends the output to the standard error output screen
- This is usually to your console screen as well
- There is a way to redirect these error messages to something else, for instance a file

Summary

- `std::cout` allows us to output to the console screen
- We use the insertion operator between each type of output
- We have two ways to get a new line
- `std::cout` and `std::cerr` are part of the `iostream` library
- We can format our output

Sample Code

- Console Output
  - `output.cpp`

Basic Console Input

Input – `std::cin`

- `std::cin` is used to get input from the console
- Our arrows go the opposite way of `std::cout`

```
std::cin >> number_of_languages;
```

  `>>` is called the **extraction operator**

- You can get more than one variable in a single `std::cin` statement

```
std::cin >> num_1 >> num_2;
```

Input – How it Works

- The program waits at **`std::cin`** statement for input to be entered
- It sets the first variable equal to the first value typed, second to second, etc.
- Does not read input until the user presses the Return key (Enter)
- To do multiple input you must separate your numbers with a space

Summary

- We use **`std::cin`** to get input from the console
- We can use one statement to get multiple input
- The extraction operator separates each variable

Sample Code

- Console Input
    - `input.cpp`

Review

- Increment / Decrement
- Type Casting
- Console Output
- Console Input