

# I/O, Variables, and Operators

---

Chantilly Robotics (Team 612)

# Console input and output

- `#include <iostream>` must be included
- Use `std::cout` to output strings
- Use `std::cin` or `getline(cin, i)`\*

\*`i` is a `std::string` variable


# Basic I/O Example

- Go to <http://cpp.sh>
- Run the Example program

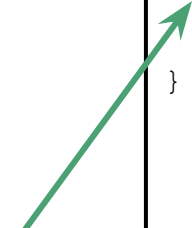
```
// Example program
#include <iostream>
#include <string>

int main()
{
    std::string name;
    std::cout << "What is your name? ";
    getline (std::cin, name);
    std::cout << "Hello, " << name << "!\n";
}
```

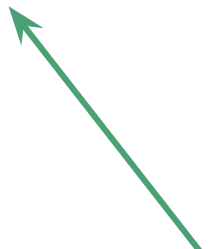
`std::string name;`  
declares a variable  
named "name"



`getline(std::cin,  
name);` gets a whole  
line from std::cin



`<<` operator inserts  
strings and variables  
into std::cout



# Console input differences

```
std::string n;  
std::cin >> n;
```

Gets user input until  
either space or newline  
character (enter)

Inputting: "Hello World!"  
Will extract "Hello" from cin and store it to  
variable n

```
std::string n;  
getline(std::cin, n);
```

Gets user input until  
newline character  
(enter)

Inputting: "Hello World!"  
Will store "Hello World!" to variable n

# Output newline character differences

```
std::cout << "Hello World!\n";
```

`\n` insert a newline character

```
std::cout << "Hello World!" <<  
std::endl;
```

`std::endl` insert a newline character and flushes the output buffer

Difference is minimal  
(`\n` has slightly better performance)

# Variables

- Used to store data
- Have types - builtin or user-defined
- Builtin types are built into the language

# Builtin Variable Types

- `int` - stores integers
- `bool` - stores true or false
- `float` - stores numbers with decimal
- `double` - double the precision of floats
- `char` - stores single characters
- `std::string` - (not a builtin) stores multiple (or a *string* of) characters

# Variable Type Example

Output: "05.1233.1415dnone"

```
// Example program
#include <iostream>
#include <string>

int main()
{
    std::string name = "none";
    int x = 0;
    float f = 5.123;
    double d = 3.1415;
    char c = 'd';
    std::cout << x << f << d << c << name;
}
```



# Operators

- Operates on variables
- You can redefine existing operators

# Arithmetic Operator List

Operator name	Syntax	Definition
Addition	<code>a + b</code>	Returns sum of a and b
Subtraction	<code>a - b</code>	Returns difference of a and b
Multiplication	<code>a * b</code>	Returns product of a and b
Division	<code>a / b</code>	Returns quotient of a and b
Modulus	<code>a % b</code>	Returns remainder of a and b
Increment	<code>++a</code> <code>a++</code>	Increases value of a by 1
Decrement	<code>--a</code> <code>a--</code>	Decreases value of a by 1

\*variable a and variable b are both integers

# Comparison Operator List

Operator name	Syntax	Definition
Is equal to	<code>a == b</code>	Returns if a is equal to b
Less than	<code>a &lt; b</code>	Returns if a is less than b
Greater than	<code>a &gt; b</code>	Returns if a is greater than
Less than or equal to	<code>a &lt;= b</code>	Returns if a is less than or equal to b
Greater than or equal to	<code>a &gt;= b</code>	Returns if a is greater than or equal to b
Is not equal to	<code>a != b</code>	Returns if a is not equal to b (false if a is equal to b)

\*variable a and variable b are both integers

# Logical Operator List

Operator name	Syntax	Definition
Negation	<code>!a</code>	Returns opposite of a
Negation	<code>not a</code>	Alternative way to write <code>!a</code>
And	<code>a &amp;&amp; b</code>	Returns true if a and b are both true
And	<code>a and b</code>	Alternative way to write <code>a &amp;&amp; b</code>
Or	<code>a    b</code>	Returns true if a or b is true
Or	<code>a or b</code>	Alternative way to write <code>a    b</code>
Conditional	<code>a ? c1 : c2</code>	If <code>a</code> , then <code>c1</code> , else <code>c2</code>

\*variable a and variable b are both booleans

# Assignment Operator List

Operator name	Syntax	Definition
Assignment	<code>a = b</code>	Assigns b to a
Addition assignment	<code>a += b</code>	Stores the sum of a and b in a ( <code>a = a + b</code> )
Subtraction assignment	<code>a -= b</code>	Stores the difference of a and b in a ( <code>a = a - b</code> )
Multiplication assignment	<code>a *= b</code>	Stores the product of a and b in a ( <code>a = a * a</code> )
Division assignment	<code>a /= b</code>	Stores the quotient of a and b in a ( <code>a = a / b</code> )
Modulus assignment	<code>a %= b</code>	Stores the remainder of a / b in a ( <code>a = a % b</code> )

\*variable a and variable b are both integers