Data Types & Expressions:

By: Mir Faraz Ali

Programming Fundamentals

Basic Data Types:

The data type specifies the size and type of information the variable will store:

Data Type	Size	Description
`boolean`	1 byte	Stores true or false values
`char`	1 byte	Stores a single character/letter/number, or ASCII values
`int`	2 or 4 bytes	Stores whole numbers, without decimals
`float`	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits
`double`	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits

C++ Numeric Data Types

- **Numeric Types:**
  - Use int when you need to store a whole number without decimals, like 35 or 1000, and float or double when you need a floating point number (with decimals), like 9.99 or 3.14515.

- **Int:**

```
int myNum = 1000;
cout << myNum;
```

- **Float:**

```
float myNum = 5.75;
cout << myNum;
```

- **Double:**

```
double myNum = 19.99;
cout << myNum;
```

C++ Boolean Data Types

- **Boolean Types:**
  - A boolean data type is declared with the bool keyword and can only take the values true or false.
  - When the value is returned, true = 1 and false = 0.

- **Example:**

```
bool isCodingFun = true;
bool isFishTasty = false;
cout << isCodingFun;  // Outputs 1 (true)
cout << isFishTasty;  // Outputs 0 (false)
```

C++ Character Data Types

- **Character Types:**
  - The char data type is used to store a single character. The character must be surrounded by single quotes, like 'A' or 'c':
  - Alternatively, if you are familiar with ASCII, you can use ASCII values to display certain characters:

- **Example:**

```
char myGrade = 'B';
cout << myGrade;
```

- **Example:**
  - char a = 65, b = 66, c = 67;
  - cout << a;
  - cout << b;
  - cout << c;

C++ String Data Types

- **String Types:**
- The string type is used to store a sequence of characters (text). This is not a built-in type, but it behaves like one in its most basic usage.
- String values must be surrounded by double quotes:

- To use strings, you must include an additional header file in the source code, the <string> library:

- **Example:**

```
string greeting = "Hello";
cout << greeting;
```

- **Example:**

```
// Include the string library
#include <string>
// Create a string variable
string greeting = "Hello";
// Output string value
cout << greeting;
```

Real life Example:

```
// Create variables of different data types
int items = 50;
double cost_per_item = 9.99;
double total_cost = items * cost_per_item;
char currency = '$';

// Print variables
cout << "Number of items: " << items << "\n";
cout << "Cost per item: " << cost_per_item << currency << "\n";
cout << "Total cost = " << total_cost << currency << "\n";
```

C++ Operators

- **C++ Operators:**
  - Operators are used to perform operations on variables and values.
  - In the example below, we use the + operator to add together two values:

    int x = 100 + 50;
- Although the + operator is often used to add together two values, like in the example above, it can also be used to add together a variable and a value, or a variable and another variable:

- **Example:**

```
int sum1 = 100 + 50;        // 150 (100 + 50)
int sum2 = sum1 + 250;      // 400 (150 + 250)
int sum3 = sum2 + sum2;     // 800 (400 + 400)
```

Arithmetic Operators

- Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

C++ Comparison Operators

- A list of all comparison operators:

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

C++ Logical Operators

- As with comparison operators, you can also test for true (1) or false (0) values with logical operators.
- Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

End Of Class