Topic3(Data Types and Variables)

C++ Variables

> Variables are containers for storing data values.

Declaring (Creating) Variables

> To create a variable, specify the type and assign it a value:

```
type variableName = value;
```

> Where type is one of C++ types(such as `int`), and variableName is the name of the variable (such as **x** or **myName)**. The **equal sign** is used to assign values to the variable.

Example

> Create a variable called **myNum** of type int and assign it the value **15**:

```
int myNum = 15
cout << myNum;
```

> You can also declare a variable without assigning the value, and assign the value later:

```
int myNum;
myNum = 15;
cout << myNum;
```

> Note that if you assign a new value to an existing variable, it will overwrite the previous value:

```
int myNum = 15;     // myNum is 15
myNum = 10;         // Now myNum is 10
cout << myNum;      // Outputs 10
```

Display Variables

> The `cout` object is used together with the `<<` operator to display variables.

> To combine both text and a variable, separate them with the `<<` operator:

```
int myAge = 35;
cout << " I am " << myAge << " years old.";
```

Add Variables Together

> To add a variable to another variable, you can use the the `+` operator:

```
int x = 5;
int y = 6;
int sum = x + y;
cout << sum;
```

C++ Declare Multiple Variables

Declare Many Variables

> To declare more than one variable of the **same type**, use a comma-separated list:

```
int x = 5, y = 6, z = 50;
cout << x + y + z;
```

One Value to Multiple Variables

> You can also assign the **same value** to multiple variables in one line:

```
int x, y, z;
x = y = z =50;
cout << x + y + z;
```

C++ Identifiers

> All C++ **variables** must be **identified** with **unique names**.

> These unique names are called **identifiers**.

> Identifiers can be short names(like x and y) or more descriptive names (age, sum, totalVolume).

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

> The general rules for naming variables are:
>
> - Names can contain letters, digits and underscores
> - Names must begin with a letter or an underscores(_)
> - Names are case sensitive (`myVar` and `myvar` are different variables)
> - Names cannot contain whitespaces or special characters like !, #, %, etc.
> - Reserved words (like C++ keywords, such as `int`) cannot be used as names

C++ Constants

> When you do not want others (or yourself) to change existing variable values, use the `const` keyword (this will declare the variable as “constant”, which means **unchangeable and read-only**):

```
const int myNum = 15;
myNum = 10;
```

> You should always declare the variable as constant when you have values that are unlikely to change:

```
const int minutesPerHour = 60;
const float PI = 3.14;
```

C++ Data Types

> In C++, there are different **types** of variables (defined with different keywords), for example:
>
> - `int` - stores integers (whole numbers), without decimals, such as 123 or -123
> - float - stores fractional numbers, containing one or more decimals. Sufficient for
> - `double` - stores floating point numbers, with decimals, such as 19.99 or -19.99
> - `char` - stores single characters, such as ‘a’ or ‘B’. Char values are surrounded by single quotes
> - `string` - stores text, such as “Hello World”. String values are surrounded by double quotes
> - `bool` - stores values with two states: true or false

```
int myNum = 5;               // Integer (whole number without decimals)
float myFloatNum = 5.99;     // Floating point number
double myFloatNum = 5.99;    // Double Floating point number (with decimals)
char myLetter = 'D';         // Character
string myText = "Hello";     // string (text)
bool myBoolean = true;       // Boolean (true or false)
```

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 number, 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

> 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;
> ```

> **NOTE :** `float` vs. `double`
> The **precision** of a floating point value indicates how many digits the value can have after the decimal point. The precision of `float` is only six or seven decimal digits, while `double` variables have a precision of about 15 digits. Therefore it is safer to use `double` for most calculations.

Scientific Numbers

> A floating point number can also be a scientific number with an “e” to indicate the power of 10:

```
float f1 = 35e3;
double d1 = 12E4;
cout << f1;
cout << d1;
```

C++ Boolean Data 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`.

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

> **NOTE :** Boolean values are mostly used for conditional testing.

C++ Character Data Types

> The `char` data type is used to store a **single** character. The character must be surrounded by single quotes, like ‘A’ or ‘C’:

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

> Alternatively, you can use ASCII values to display certain characters:

```
char a = 65, b = 66, c = 67;
cout << a;
cout << b;
cout << c;
```

> Tip : A list of all ASCII values can be found in our ASCII Table Reference.

C++ String Data 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 quoutes:

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

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

```
// Include the string library
#include<string>

// Create a string variable
string greeting = "Hello";

// Output string value
cout << greeting;
```