Variable Types

This lesson gives an overview of all the types of variables in C++ like int, bool, double, char and float

We'll cover the following

- Integers
- Char
- Floats
- Doubles
- Boolean

Integers

An *integer* is a number that does not have any decimal places. It is a *whole number*, for example, **1,2,3,4** are all integers. **4.3** is **not**. If you were to try and place the number **4.3** into an integer, the number would be truncated to **4**.

There are further different types in an integer as well. Let’s take a look at them one by one.

Below is an example of the integer type `short`.

```
// Short is normally defined as a 16-bit integer.

short myVariableName1;  //bytes and  stores from -32768 to +32767
short int myVariableName2;  //stores from -32768 to +32767
signed short myVariableName3;  // stores from -32768 to +32767
signed short int myVariableName4;  // stores from -32768 to +32767
unsigned short myVariableName5;  // stores from 0 to +65535
unsigned short int myVariableName6;  // stores from 0 to +65535
```

Another type of `int` is the **16** bit and **32** bit integers.

```
// Int is guaranteed to be 16-bit, but modern implementations use 32-bit for an int.

int myVariableName7;  // stores from -32768 to +32767
signed int myVariableName8;  // stores from -32768 to +32767
unsigned int myVariableName9;  // stores from 0 to +65535
```

The integers can also be of the type `long`.

```
// Long is a 32-bit number.

long myVariableName10;  // stores from -2147483648 to +2147483647
long int myVariableName11;  // stores from -2147483648 to +2147483647
signed long myVariableName12;  // stores from -2147483648 to +2147483647
signed long int myVariableName13;  // stores from -2147483648 to +2147483647
unsigned long myVariableName14;  // stores from 0 to +4294967295
unsigned long int myVariableName15;  // stores from 0 to +4294967295
```

Now we can attribute reasons to the *ranges* of `int`, `long` and so on.

For `int` :

```
2^16 = 65536
```

Now, this is the total range of a variable.

- Dividing by **2** we get, **32768**.

So **-32768** to **32767** (it should have been **32768** but **1** place is taken by **0**).

Now the size of `long` is **4** bytes **(32 bits)**.

- **So the *range* is** 2^32.

What is the difference between a `long` and a `signed long int`? In my mind, the only difference is **12** extra keystrokes. Pick one that works for you.

Char

A `char` is an **8-bit** integer. This means that an *unsigned* `char` can store between **0** and **255**, and a *signed* `char` can store between **-128** and **127**. *Unsigned* chars are commonly used to store text in **ASCII** format. A `char` can be initialized to hold either a number or a character, but it will store **only** the *ASCII* value.

```
#include <iostream>
using namespace std;

int main()
{
  char myChar='A';
  char myOtherChar=65;
  cout << "Value of myChar is: "<< myChar<<endl;
  cout << "Value of myOtherChar is: "<< myOtherChar<<endl;
  return 0;
}
```

Both characters that I have just initialized would be equal.

- The number **65** is the *ASCII* code for the letter **'A'**

so both characters would contain the **8-bit** value of **65**, or the letter **'A'**.

> **Note:** ASCII is a system where a numerical value is assigned to every character you can think of. For a complete conversion chart visit http://ascii-code.com/

Floats

Floats are *floating* point numbers with a storage size of **4** *bytes*, which means that these numbers can hold *decimal* places. This allows us to store numbers such as **"8.344"** and **"3432432653.24123"**.

```
#include <iostream>
using namespace std;
int main()
{
  float myFloat;  // Creates a floating point variable
  myFloat = 8.3;  // Stores 8.3 in the new variable
  cout<< "Value of myFloat is: "<< myFloat << endl;
  return 0;
}
```

Floating point numbers have a *fixed* size in memory. This means that a *single* `float` **cannot** possibly precisely store all of the *decimal* values in the real number system.

> **Note:** The `float` data type usually stores only a good approximation of a decimal value, not the exact value.

Doubles

Doubles are like *"floats"*, which means they can store decimal places. Doubles can generally store **more** information than a standard `float`. Their *storage* size is of **8** *bytes*.

```
#include <iostream>
using namespace std;
int main()
{
  double myDouble;   // Created myDouble
  myDouble = 8.78;   // Stores 8.78 in myDouble
  cout << "Value of myDouble is: "<< myDouble <<endl;
  return 0;
}
```

As noted with the `float` data type, `double` usually stores only an approximation of **exact** *decimal* values (albeit, usually a higher precision approximation than the smaller `float` data type).

Boolean

The `bool` (boolean) type is a **1-byte** data type that is either *true* or *false*. A `true` being any number *other* than **zero** and `false` being **zero**. The `true` keyword uses the value **1** to assign `true`.

```
#include <iostream>
using namespace std;
int main()
{
  bool canJump = false;
  bool canDo = true;
  cout<< "Value of canJump is: "<< canJump<<endl;
  cout<< "Value of canDo is: "<< canDo<<endl;
  return 0;
}
```

Now that you're familiar with the variable types let's look at interesting operations in C++ in the upcoming chapter!