Variables and Data Types in C++

Dr Ian Cornelius

Hello

Hello (1)

Learning Outcomes

1. Understand what a variable is, and the various data types in C++
2. Demonstrate the ability to declare variables of various data types

Variables

Variables (1)

- You should remember what a variable is from the first year programming module
- To recap:
  - a variable is a reserved memory location to store values of a particular data type
  - consist of two parts: a *name* and a *value*
- There are a few rules to follow when naming your variables:
  1. Names must start with a letter or an underscore
  2. Names can only contain alphanumeric characters and underscores (`A-z`, `0-9`, and `_`)
- If you need to supply a comment to explain a name, then it does not reveal its true intent
  - if this is the case, then you may want to reconsider renaming your variable
- **Note**: that variable names are case-sensitive:
  - i.e. `module`, `Module` and `MODULE` are all different variables (and memory locations)

Variables (2)

Declaring a Variable

- Variable declaration in C++ is different compared to Python
- C++ requires you to declare the data type at the beginning of declaration
  - **note**: the semicolon (`;`) - this is important!
- Variables are assigned values using the equal symbol (`=`)
  - the operand to the left of the `=` is the *name* of the variable
  - the operand to the right of the `=` is the *value* of the variable

Python Declaration

```
aVariable = "String Data Type"
anotherVariable = 1.0
```

C++ Declaration

```
string aVariable = "String Data Type";
float anotherVariable = 1.0;
```

Data Types

Data Types (1)

- Data types represent the data that is stored in the memory of the computer
- There are two types of data types:
  - **basic**: these are native to C++, they consist of:
    - Boolean
    - Character
    - String
    - Numeric
    - Vectors/Arrays (**Discussed in Week 8**)
    - Maps (**Discussed in Week 8**)
  - **user-defined**: these are created by the programmer in the forms of classes and objects
    - classes and objects are discussed in **Week 9**

Data Types (2)

Boolean

- Declared using the `bool` keyword
- Represents boolean values which are either `true` or `false`
  - **note**: these are case-sensitive and are all lower-case compared to Python
- When returned, the value of a boolean is numeric
  - `true` = `1`
  - `false` = `0`

```cpp
#include <iostream>
bool boolExample1 = true;
bool boolExample2 = false;
int main() {
    std::cout << "boolExample1 -> " << boolExample1 << std::endl;
    std::cout << "boolExample2 -> " << boolExample2 << std::endl;
    return 0;
}
```

```
boolExample1 -> 1
boolExample2 -> 0
```

Data Types (3)

Character

- Declared using the `char` keyword
- Stores a **single** character
  - must be surrounded by a set of single quotes (`'`)

```cpp
#include <iostream>
char charExample1 = 'A';
char charExample2 = 'b';
char charExample3 = 'abc';
int main() {
    std::cout << "charExample1 -> " << charExample1 << std::endl;
    std::cout << "charExample2 -> " << charExample2 << std::endl;
    std::cout << "charExample3 -> " << charExample3 << std::endl;
    return 0;
}
```

```
charExample1 -> A
charExample2 -> b
charExample3 -> c
```

Data Types (4)

String

- Declared using the `string` keyword
- Stores **multiple** characters, otherwise referred to as a *sequence of characters*
  - must be surrounded by a set of double quotes (`"`)
- Not *strictly* an in-built data type
  - requires an additional module to include called `string`

```cpp
#include <iostream>
#include <string>
std::string strExample1 = "This is with numbers: 1, 2, 3, 4.";
std::string strExample2 = "A";
int main() {
    std::cout << "strExample1 -> " << strExample1 << std::endl;
    std::cout << "strExample2 -> " << strExample2 << std::endl;
    return 0;
}
```

```
strExample1 -> This is with numbers: 1, 2, 3, 4.
strExample2 -> A
```

Numeric Data Types

Numeric Data Types (1)

- There are three numeric data types in C++:
  - Integers
  - Floats
  - Doubles

Numeric Data Types (2)

Integer

- Declared using the `int` keyword
- Represents an integer number, i.e., a number without a decimal point
- The integer data type represents both huge negative and positive integer numbers
  - a range from `-2147483648` to `2147483648`

```
#include <iostream>
int intExample1 = 1;
int intExample2 = -1024;
int main() {
    std::cout << "intExample1 -> " << intExample1 << std::endl;
    std::cout << "intExample2 -> " << intExample2 << std::endl;
    return 0;
}
```

```
intExample1 -> 1
intExample2 -> -1024
```

Numeric Data Types (3)

Float

- Declared using the `float` keyword
- Represents a floating point number, i.e., a number with a decimal point

```cpp
#include <iostream>
#include <iomanip>
float floatExample1 = 1.584;
float floatExample2 = 1024.9876453;
int main() {
    std::cout << "floatExample1 -> " << floatExample1 << std::endl
    std::cout << "floatExmaple2 -> " << floatExample2 << std::endl
    std::cout << "floatExmaple2 -> " << std::setprecision(20) <<
    return 0;
}
```

```
floatExample1 -> 1.584
floatExmaple2 -> 1024.99
floatExmaple2 -> 1024.9876708984375
```

Numeric Data Types (4)

Double

- Declared using the `double` keyword
- Represents a floating point number, i.e., a number with a decimal point

```cpp
#include <iostream>
#include <iomanip>
double dblExample1 = 1.584;
double dblExample2 = 1024.9876453;
int main() {
    std::cout << "dblExample1 -> " << dblExample1 << std::endl;
    std::cout << "dblExample2 -> " << dblExample2 << std::endl;
    std::cout << "dblExmaple2 -> " << std::setprecision(20) << db
    return 0;
}
```

```
dblExample1 -> 1.584
dblExample2 -> 1024.99
dblExmaple2 -> 1024.9876452999999401
```

Numeric Data Types (5)

Floats vs. Doubles

- The difference between the two is the floating point value
  - indicates the number of values *after* the decimal point
- Float can have seven digits
  - i.e. `10.1234567`
- Double can have fifteen digits
  - i.e. `10.1234567890123456`

Example: Float Division

```cpp
#include <iostream>
#include <iomanip>
float floatExample1 = 1.0 / 81;
float floatExample2 = 0.0;
int main() {
    for (int i = 0; i < 729; ++i) {
      floatExample2 += floatExample1;
    }
    std::cout << "floatExample2 -> " << std::setprecision(7) << f
    std::cout << "floatExample2 -> " << std::setprecision(15) <<
    return 0;
}
```

```
floatExample2 -> 9.000023
floatExample2 -> 9.00002288818359
```

Example: Double Division

```cpp
#include <iostream>
#include <iomanip>
double dblExample1 = 1.0 / 81;
double dblExample2 = 0;
int main() {
    for (int i = 0; i < 729; ++i) {
      dblExample2 += dblExample1;
    }
    std::cout << "dblExample2 -> " << std::setprecision(7) << dbl
    std::cout << "dblExample2 -> " << std::setprecision(15) << db
    return 0;
}
```

```
dblExample2 -> 9
dblExample2 -> 8.99999999999996
```

Goodbye

Goodbye (1)

Questions and Support

- Questions? Post them on the **Community Page** on Aula
- Additional Support? Visit the Module Support Page
- Contact Details:
  - Dr Ian Cornelius, ab6459@coventry.ac.uk