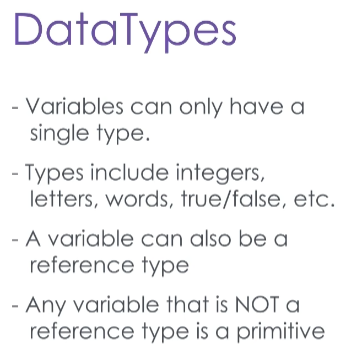
One of the most basic features needed by any programming language is the ability to save data for future use. The structures we use to store data are typically referred to as fields, properties, or variables.



C# is a strongly typed language, which means that every variable in C# can only hold a particular type of data.

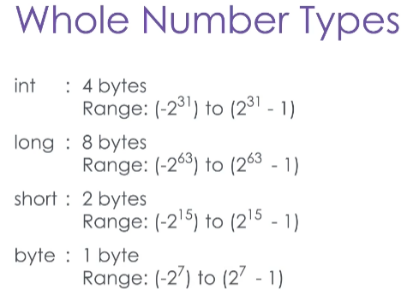
For example, it might hold a whole number – an integer.

Or it might hold a single character of data, or a true/false value, or a line of text.

A variable might also be a reference to an Object in memory – you’ll recall that’s a specific instance of another class.

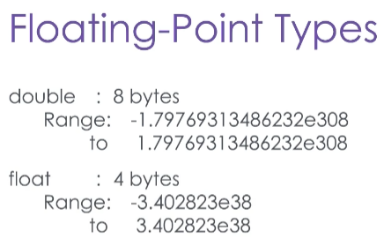
Any data type that is not a reference to an object is called a primitive data type.

Different types of data require different amounts of memory to represent them, and so variables of different types reserve different amounts of memory to store data of that type.

For example, whole numbers in C# are usually stored in a 32-bit int variable – short for Integer.

However, if 32 bits isn’t enough to express a number as large as we need to store, we can also use the 64-bit long type.

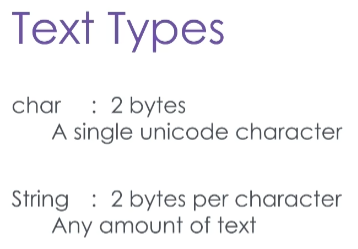
On the other hand, if we’re trying to be really efficient, we can store smaller whole numbers in a 16-bit short variable, or even an 8-bit byte.



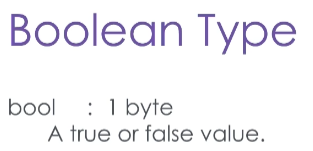
Decimal values are typically stored in a double, which is also 64-bits long.

Without getting into the nitty gritty of binary arithmetic, the more bits you use to store decimal data, the more precise those decimal values can be.

If you don’t need as much precision, or if you don’t need to store as large a number as a double, you can use the 32-bit float type.

A single text character can be stored in a char type, which takes 16 bits of memory. The range of values that can be assigned to a character variable is based on the Unicode Standard, which contains over 100,000 different types of characters, including the alphabets of most languages, digits, and various other symbols.

Of course, you can also store any amount of text as a string. Unlike most other types of variables, strings do not take up a predetermined amount of space – the longer the string is, the more memory will be reserved to hold it. You can store just about any type of text in a string.

The last type of primitive data you’ll work with in C# is Boolean data, which is shortened to the keyword bool.

Boolean data is a simple true/false value, and it occupies only 8-bits of memory.

Now let’s look at an example which demonstrates these data types.

using System.Collections.Generic;

using System.Ling;

Using System.Text;

using System.Threading.Tasks;

namespace DataYpes

{

class Program

{

static void Main(string[] args)

{

int i = 1;

int onlineUsers;

onlineUsers = 2;

i = 50;

double d = 1.5;

double accountBalance = 120.35;

d = 50.6;

char c = ‘a’;

char dollar = ‘$’;

string s = “hello world”;

string welcomeMessage = “Welcome to the data types demo.”;

string emptyString = “”;

string oneCharacter = “a”;

bool b = true;

bool activeBankAccount = false;

activeBankAccount = b;

}

}

}

Let’s take a look at the first line of our Main method, where I’ve declared a variable called ‘i’ of type ‘int’, which is assigned the value 1 with the equals operator. On the line below, I’ve declared another integer variable with a more descriptive name, “onlineUsers”. In the old days, resources were so tight that even variable names were shortened to save memory, but modern computers are not so constrained, and Intellisense which makes it quicker and easier to use properly declared variables.

To demonstrate, add a new line after declaration of ‘online users’ and type the letters “o, n”. You’ll notice that Intellisense immediately suggests the variable “onlineUsers”. Just press enter and the variable name is typed for you.

Using the equals operator, we’re going to change the value to 2 and don’t forget to add the semicolon at the end of the line. On the next line let’s reassign the value of variable “i” to 50.

Moving on to decimal numbers, I’ve declared and assigned 2 variables. I’ve created a new double called “d” and assigned it the value of 1.5. Then I’ve declared the variable “accountBalance”, also a double, and assigned it value 120.35. On the line below “accountBalance”, let’s change the value of variable “d” by typing the letter “d” and “=”, then assigning it the value 50.6. Once again, don’t forget the semicolon.

For the next data types example, I’ve declared 2 character variables called “c” and ”dollar”. And I’ve assigned the literal characters ‘a’ and ‘$’ sign using the single quotes or apostrophes.

For the string data types I’ve declared two strings. The first one, variable “s”, is assigned the literal string “hello world” and the other string, “welcomeMessage”, is assigned “Welcome to the data types demo.” Now I’m declaring a new string called “emptyString” and assigning it the empty double quotes to signify an empty string. You can also declare a strings to contain a single character like where I assign the letter “a” to the new variable “oneCharacter”

Let’s take a moment to discuss common practices for naming variables.

Be aware that variable names cannot have spaces, and it’s also standard practice that they be written in lower camel case. They should start with a lower case first letter, and then every first letter of the following words in the name is capitalized. This makes it easier to distinguish at first glance, the difference between a variable and a method or a class.

The last examples are Boolean data types, and if you recall the values are either ‘true’ or ‘false. Now I’ve declared two variables, “b” and “activeBankAccount”, and assigned them the values true and false, respectively. On the next line, I’m going to change the value of “activeBankAccount” to the value of “b” using the equals operator once again.

To complete this demonstration, I’m going to paste several lines of code I’ve written earlier to finish this console application. You’ll remember the “WriteLine” method simply writes a string to the console. The “Write” method also writes a string to the console, but the cursor doesn’t move to the next line after the string is printed. Now I’m going to run this program in debug mode, by pressing the “F5” shortcut key.

When the console application appears it will print the welcome message I declared earlier, then display the account active status which was changed to the value true, followed by the account balance message using the “dollar” variable character and the “accountBalance” double, also declared above.