**Data Types:**

* C# is a type-safe language.
* 2 types of Data types:
  + Value Types:
    - variable contains object with the value.
    - Primitives, struct, enum,
  + Reference Types:
    - contains reference to an object.
    - If values are changed using one variable, it effects other too.
    - Class, object, interface, string, array, delegate
* sizeof() is used to get the size(in bytes) occupied by a datatype in the memory
* **Primitive(Basic/Predefined) Datatypes:**
  + byte (unsigned)
    - occupies 1 byte (8 bits)
    - range: 0 to 28-1
    - System.Byte
  + sbyte (signed)
    - occupies 1 byte (8 bits)
    - range: -27  to 27-1
    - System.Sbyte
  + char (Unicode character)
    - occupies 2 bytes (16 bits)
    - range: U+0000 to U+ffff
    - System.Char
  + short (signed)
    - occupies 2 bytes (16 bits)
    - range: -215  to 215-1
    - System.Int16
  + ushort (unsigned)
    - occupies 2 bytes (16 bits)
    - range: 0 to 216-1
    - System.Int16
  + int (signed)
    - occupies 4 bytes (32 bits)
    - range: -231  to 231-1
    - System.Int32
  + uint (unsigned)
    - occupies 4 bytes (32 bits)
    - range: 0 to 232-1
    - System.UInt32
  + long (signed)
    - occupies 8 bytes (64 bits)
    - range: -263  to 263-1
    - System.Int64
  + ulong (unsigned)
    - occupies 8 bytes (64 bits)
    - range: 0 to 264-1
    - System.UInt64
  + bool
    - occupies 1 byte (8 bits)
    - System.Boolean
  + float
    - occupies 4 bytes (32 bits)
    - range: ±1.5 x 10−45 to ±3.4 x 1038
    - System.Single
  + double
    - occupies 8 bytes (64 bits)
    - range: ±5.0 × 10−324 to ±1.7 × 10308
    - System.Double
  + decimal
    - occupies 16 bytes (128 bits)
    - range: ±1.0 x 10-28 to ±7.9228 x 1028
    - System.Decimal
* Enumeration:
  + Constants declared under a single name using keyword “enum”
  + By default, the keys will be 0-index based.
  + We can set our own indices to the constants.
* Void:
  + When a method does not return anything, void will be used as return type
  + It can’t be a parameter.
* Command line arguments:
  + Will be taken as string array (if any) in Main method arguments.
* Object:
  + Can store any data type.
  + Boxing: converting a datatype into object
  + Unboxing: converting object into its data type
  + We should know the original type of data before unboxing.
  + Eg:
    - Object o = 10;
    - Int i = (int) o;
* Var:
  + Used to declare implicit variable (when we don’t know the type of the data)
  + Eg: var num = “98”;
  + It must be initialized during declaration, throws error otherwise.
  + It takes value as string when read from console.
  + Otherwise, it will typecast data implicitly during compile time.
* Dynamic:
  + Similar to Var but need not be initialized.
  + It typecasts data during runtime.
  + It takes value as string when read from console.
  + Eg: dynamic d; d=9;
* Struct:
  + Struct can have data and methods.
  + Does not have constructor, can’t be inherited.
  + Similar to class but Value type data
  + Setters can be used to set data since it does not support constructors to initialize data with default values.
  + Object declaration is similar to class, using “new” keyword.
  + “.” operator is used to access the data and methods of struct.
* Class:
  + Class can have data and methods.
  + Constructors and inheritance are supported.
  + Reference type
  + Data items will be initialized with default values using default constructors.
  + Objects are declared using “new” keyword.
  + “.” Operator is used to access data and methods of class.
* Delegate:
  + Declaring methods with “delegate” keyword.
  + The delegate method will have declaration only.
  + When creating delegate object, we send method with actual implementation as parameter.
  + By using this, we can encapsulate methods.
* Searching in C#:
  + Search for an item in array
  + Internal searching: searching in the main memory.
  + External searching: searching in the secondary memory.
  + 2 popular techniques:
    - Linear/Sequential search:
      * searching data in a linear manner.
      * Used for arrays with randomized data.
      * Complexity: O(n)
    - Binary search:
      * Used for searching in sorted array.
      * Complexity: O(log n)