What is Variable?

A Variables is a named storage location that holds a value or data. These values can change during the execution of a program, hence the term "variable." Variables are essential for storing and manipulating data in computer programs. A variable is the basic building block of a program that can be used in expressions as a substitute in place of the value it stores.
Example: z = x + y
This is an example of programming expression x, y and z are variables. Variables can represent numeric values, characters or character strings. In a program every, variable has:

- Name (Identifier)
- Data Type
- Size
- Value

What is Data type?
A data type specifies the size and type of variable values. It is important to use the correct data type for the corresponding variable; to avoid errors, to save time and memory, but it will also make your code more maintainable and readable. The most common C# data types are:

C# Data Types

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for
		storing 15 decimal digits
bool	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter, surrounded by single quotes
string	2 bytes per character	Stores a sequence of characters, surrounded by double quotes

Data Types

1. System-defined data types (also called Primitive data types)
2. User-defined data types

System-defined data types

Data types that are defined by system. The primitive data types provided by many programming languages are: int, float, char, double, bool, etc. The number of bits allocated for each primitive data type depends on the programming languages, the compiler and the operating system.

User defined data types

If the system-defined data types are not enough, then most programming languages allow the users to define their own data types

- e.g., structures in C/C + + and classes in Java

Data Structures

A data structure is a way of organizing and storing data in a computer so that it can be accessed and used efficiently. It refers to the logical or mathematical representation of data, as well as the implementation in a computer program. A data structure is a specialized format for organizing, processing, retrieving and storing data.

Classification of Data Structures

1. **Linear data structures:**
   A linear data structure is a structure in which the elements are stored sequentially, and the elements are connected to the previous and the next element. As the elements are stored sequentially, so they can be traversed or accessed in a single run. The implementation of linear data structures is easier as the elements are sequentially organized in memory. The data elements in an array are traversed one after another and can access only one element at a time.
   e.g., Array, Queue, Stack, Linked List

2. **Non - linear data structures:**
   A non-linear data structure is also another type of data structure in

which the data elements are not arranged in a contiguous manner. As the arrangement is nonsequential, so the data elements cannot be traversed or accessed in a single run. In the case of linear data structure, element is connected to two elements (previous and the next element), whereas, in the non-linear data structure, an element can be connected to more than two elements.

e.g., Trees and Graphs

	Linear Data structure	Non-Linear Data structure
Basic	In this structure, the elements are arranged sequentially or linearly and attached to one another.	In this structure, the elements are arranged hierarchically or non-linear manner.
Types	Arrays, linked list, stack, queue are the types of a linear data structure.	Trees and graphs are the types of a non-linear data structure.
implementation	Due to the linear organization, they are easy to implement.	Due to the non-linear organization, they are difficult to implement.
Traversal	As linear data structure is a single level, so it requires a single run to traverse each data item.	The data items in a non-linear data structure cannot be accessed in a single run. It requires multiple runs to be traversed.
Arrangement	Each data item is attached to the previous and next items.	Each item is attached to many other items.
Levels	This data structure does not contain any hierarchy, and all the data elements are organized in a single level.	In this, the data elements are arranged in multiple levels.
Memory utilization	In this, the memory utilization is not efficient.	In this, memory is utilized in a very efficient manner.

Time complexity	The time complexity of linear data structure increases with the increase in the input size.	The time complexity of non- linear data structure often remains same with the increase in the input size.
Applications	Linear data structures are mainly used for developing the software.	Non-linear data structures are used in image processing and Artificial Intelligence.

Abstract Data Types (ADT)

- A set of **data** values and associated **operations** that are precisely specified independent of any particular implementation.

Commonly used ADTs include: Linked Lists, Stacks, Queues, Priority Queues, Binary Trees, Dictionaries etc.

- For example, stack uses LIFO (Last-In-First-Out) mechanism while storing the data in data structures. The last element inserted into the stack is the first element that gets deleted. Common operations of it are: creating the stack, pushing an element onto the stack, popping an element from stack, finding the current top of the stack, finding number of elements in the stack, etc.
- Some operations in Queue are the following:
  - isFull(), This is used to check whether queue is full or not
  - isEmpty(), This is used to check whether queue is empty or not
  - insert(x), This is used to add x into the queue at the rear end
  - delete(), This is used to delete one element from the front end of the queue
  - size(), this function is used to get number of elements present into the queue