Data Types and Variables

By Dhananjay Masal– CODEMIND Technology

Contact us +91 9890217463

Data Types

- Why need data type in C#?
  - Used to store the data temporarily.

- Size of the memory location.

- Range of data that can be stored inside memory location

- legal operations that can be performed on that memory location

- What types of result come out from an expression when types are used inside that expression.

C# Data Types

We have three types of Data types:

1. Value data type (Primitive)
2. Pointer data type
3. Reference data type (Non-Primitive)

1. Value data type

- Stores value directly
- Derived from class System.ValueType

Types of Value data type:

1. Pre-defined - e.g. int, boolean, float etc.
2. User defined - e.g. structure, Enum

2. Reference Data Type

- Used to store reference of variable

Types of Value data type:

1. Pre-defined - e.g. Object, string, dynamic.
2. User defined - e.g. classes, interface

Built-in Data Types in C#

1. Boolean types - true/false.
2. Integral Types - sbyte, byte, short, ushort, int, uint, long, ulong, char
3. Floating Types - float, double
4. Decimal
5. String

Why String type immutable:

- Strings are immutable, which means we are creating new memory every time instead of working on existing memory. [System.String]
- Using same memory location & keep an appending/modifying the stuff to one instance [eg String Builder]
- Why string immutable - for thread safety.

Questions:

1. Immutable vs Mutable?
2. Why string is immutable?

Static & non-static members

1. Static members: does not require an instance for initialization or execution are know as static members.
2. Non-Static members: the member which require an instance of class for both initialization & execution are know as non-static members.

Static & non-static members eg

```
 Int p =100;                // non-static variables
Static int q =50;       // static variables
Static void Main(){
        int r = 100;      // static variables
}
```

Static & non-static methods:

- If we declare method using the static modifier then it is called static method else it is non-static.
- Cannot consume non-static members directly within static

Rules while working with static & non-static:

1. Non-static to static: Can be consumed only by using object of class.
2. Static to static: Consumed directly or by using class name
3. Static to non-static: Consumed directly or by using class name.
4. Non-static to non-static: Consumed directly or by using “this” keywords

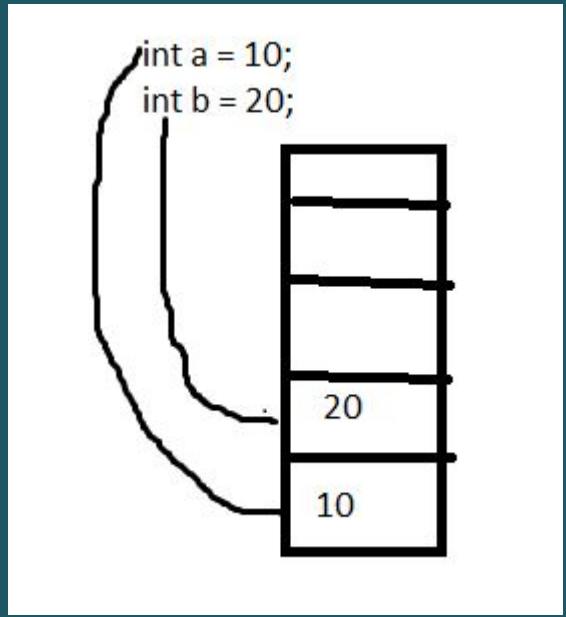
Static Class in C#

- Class which created by using static modifiers
- Only Static members in it.
- Static class cannot be instantiated, cannot use new operator to create variable of class.
- Is **Sealed** by default

Stack vs Heap

Stack :

1. It is an array of memory.
2. It is LIFO data structure.
3. Value of variable storing in stack.
4. Value type - int, long,double, bool etc
5. Static memory allocations
6. Memory deallocate when scope ends.

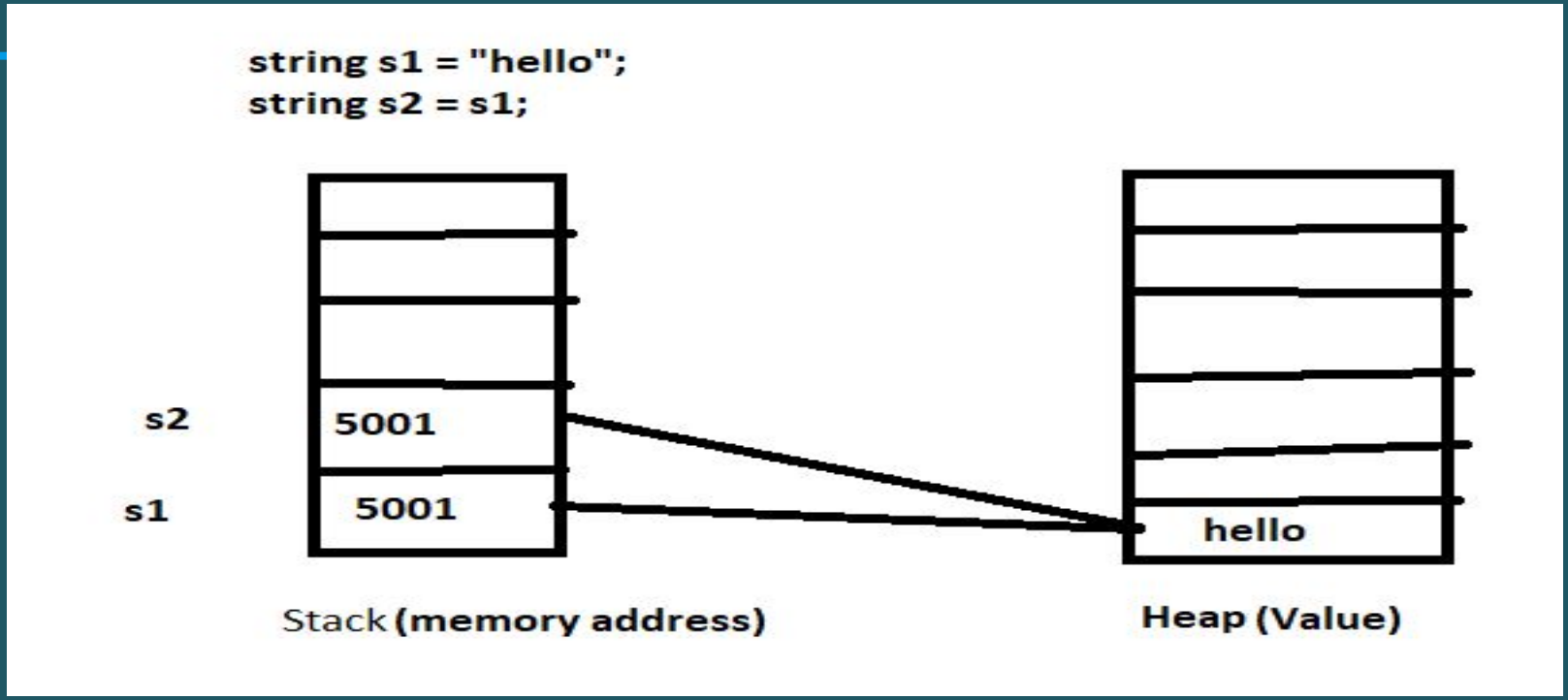


Stack vs Heap

Heap:

1. It is an array of memory where chunks are allocated to store certain kinds of data object.
2. Reference type store on heap..
3. Reference type - string, class, object etc
4. Dynamic memory allocations.
5. Heap clear when **GC.Collects()/**Garabage collector handle it.

Heap Eg.


string s1 = "hello";
string s2 = s1;
s2
5001
s1
5001
hello
Stack (memory address)
Heap (Value)

Data type conversion

1. Implicit conversion: - It is done by **compiler** when
    a. When there is no loss of information if conversion done.
    b. If there is no possibility of throwing exceptions during the conversion

    Eg. Converting int to float

    Int a =10;

    Float b = a; //10.0

Data type conversion

1. Explicit conversion: - It is done by **manually** when
    a. If there is possibility of throwing overflow exceptions during the conversion
    b. Using **cast()** operator

    Eg. Converting float to int

    Float a =10.5;

    Int b = (int)a;

Methods

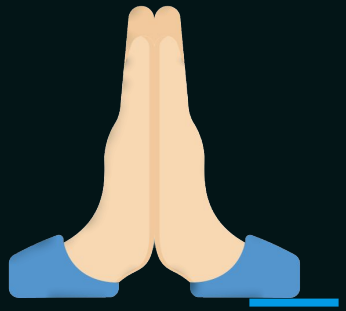
A method is a group of statements that together perform a task.

```
<Access Specifier> <Return Type> <Method Name>(Parameter List) {
   Method Body
}
```

Methods

1. Access Specifier − This determines the visibility of a variable or a method from another class.
2. Return type − A method may return a value. The return type is the data type of the value the method returns. If the method is not returning any values, then the return type is void.
3. Method name − Method name is a unique identifier and it is case sensitive. It cannot be same as any other identifier declared in the class.
4. Parameter list − Enclosed between parentheses, the parameters are used to pass and receive data from a method. The parameter list refers to the type, order, and number of the parameters of a method. Parameters are optional; that is, a method may contain no parameters.
5. Method body − This contains the set of instructions needed to complete the required activity.

Thank You



Success is not a milestone, it's a journey. And we have vowed to help you in yours.

CODEMIND
Your Success Partner

www.codemindtechnology.com