NANYANG POLYTECHNIC School of Information Technology

Table of Contents

Data Types

Unlike most programming languages, C# makes use of a unified type system. All C# types, including primitive types such as integer and double, it inherits from a single root called object type. Values of any type can be stored, transported and operated upon in a consistent manner.

int	Integer values like 1234, 10000
double	64-bit floating-point, 3.145644
float	Floating-point number, 3.1454
string	Set of characters, “welcome”
byte	8-bit unsigned integer
Char	16-bit unicode character, ‘A’
long	64-bit signed integer, -9.0789
decimal	High precision decimal numbers
bool	True or false boolean values

String Data type conversion

AsInt(), IsInt()	Convert string into integer Check if the input is int
AsFloat(), IsFloat()	Convert string into float, Check if the input is float
AsDecimal(), IsDecimal()	Convert string into decimal, Check if the input is decimal
AsDateTime(), IsDateTime()	Convert string into date-time type, Check if the input is date-time type
AsBool(), IsBool()	Convert string into bool, Check if the input is bool

Comparison Operators

Comparison operators, just like python, are commonly used to compare variables to one another and perform general evaluations.

==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Arithmetic Operators

Just like comparison operators, there are arithmetic operators, which are used to perform common mathematical functions on variables. The syntax is largely similar to python.

+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	x++

--	Decrement	Decreases the value of a variable by 1	x--

Logical Operators

Logical operators are necessary to perform decisions based on truth tables and logic circuits. They result in boolean outputs.

&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	X < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

String Operations

In C#, strings are objects, which have properties and methods. A variety of string properties can be discovered or called upon using string operations and methods.

Clone()	Make clone of string	str2 = str1.clone()
CompareTo()	Compares two strings and returns integer value as output. It returns 0 for true and 1 for false	str2.CompareTo(str1)
Contains()	Checks whether specified character or string exists or is not in the string value	str2.Contains(“hack”)
EndsWith()	Checks whether the specified character is the last character of the string or not.	str2.EndsWith(“lo”)
Equals()	Compares two strings and returns boolean value true as output if they are equal, false	str2.Equals(str1)

	if not	
GetHashCode()	Returns HashValue of a specified string	str1.GetHashCode()
GetType()	Returns the System.Type of the current instance	str1.GetType()
GetTypeCode()	Returns the System.TypeCode for class System.String	str1.GetTypeCode()

String Methods

IndexOf()	Returns the index position of first occurrence of specified character	str1.IndexOf(“:”)
ToLower()	Converts string into lower case based on rules of the current culture	str1.ToLower()
ToUpper()	Converts string into upper case based on the rules of the current culture	str1.ToUpper()
Insert()	Insert the string or character in the string at the specified position.	str1.Insert(0,”Welcome”) str1.insert(“i”,”Thank You”)
IsNormalized()	Check whether this string is in unicode normalization form	str1.IsNormalized()
LastIndexOf()	Returns the index position of last occurrence of specified character	str1.LastIndexOf(“T”)
Length()	Returns length of string	str1.Length()
Remove()	Deletes all the characters from beginning to specified index position	str1.Remove(i)
Replace()	Replaces the specified character with another	str1.Replace(“a”,”e”)

Split()	This method splits the string based on a specified value	Str1 = “Good morning and Welcome”; String sep = {“and”}; strArray = str1.Split(sep, StringSplitOptions.None);
StartsWith()	Checks whether the first character of string is same as specified character	str1.StartsWith(“H”)
Substring()	This method returns a substring	str1.Substring(1,7)
ToCharArray()	Converts string into char array	str1.ToCharArray()
Trim()	It removes extra whitespaces from beginning and ending of string	str1.Trim()

Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. A variety of array properties can be discovered or called upon using array operations and methods.

Find()	Searches for an element that matches the conditions defined by the specified predicate, and returns the first occurrence within the array	Array.Find(arrVal, <matching pattern>)
FindAll()	Retrieves all the elements that match the conditions defined by the specified predicate	Array.FindAll(arrVal, <matching pattern>)

FindIndex()	Searches for an element that matches the conditions defined by a specified predicate, and returns the zero-based index of the first occurrence within an array or a portion of it	Array.FindIndex(arrVal, <matching pattern>)
FindLast()	Searches for an element that matches the conditions defined by the specified predicate, and returns the last occurrence within the entire array	Array.FindLast(arrVal, <matching pattern>)
FindLastIndex()	Searches for an element that matches the conditions defined by the specified predicate, returns the zero-based index of the last occurrence within an array or a portion of it	Array.FindLastIndex(arrVal, <matching pattern>)
ForEach()	Loops through each element of the array and performs the specified action	Array.ForEach(arrVal, Action)

Conditional statements

C# has the following conditional statements;
Use if to specify a block of code to be executed, if a specified condition is true
Use else to specify a block of code to be executed, if the same condition is false
Use else if to specify a new condition to test, if the first condition is false
Use switch to specify many alternative blocks of code to be executed

if-else	if (true) {...} else if (true) {...} else {...}

switch	switch (var)
	{
	case 1: break;
	case 2: break;
	default: break;
	}

for	for (int i = 0; i <= len; i++) {...}
foreach-in	foreach (int item in array) {...}
while	while (true) {...}
do…while	do {...} while (true);