C# - Ultimate Guide - Beginner to Advanced | Master class

Section 27 – IO, Serialization, Encoding

Number Systems

In mathematics, a 'Number system' provides a set of digits for counting.

Decimal: Base 10

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Octal: Base 8

0, 1, 2, 3, 4, 5, 6, 7

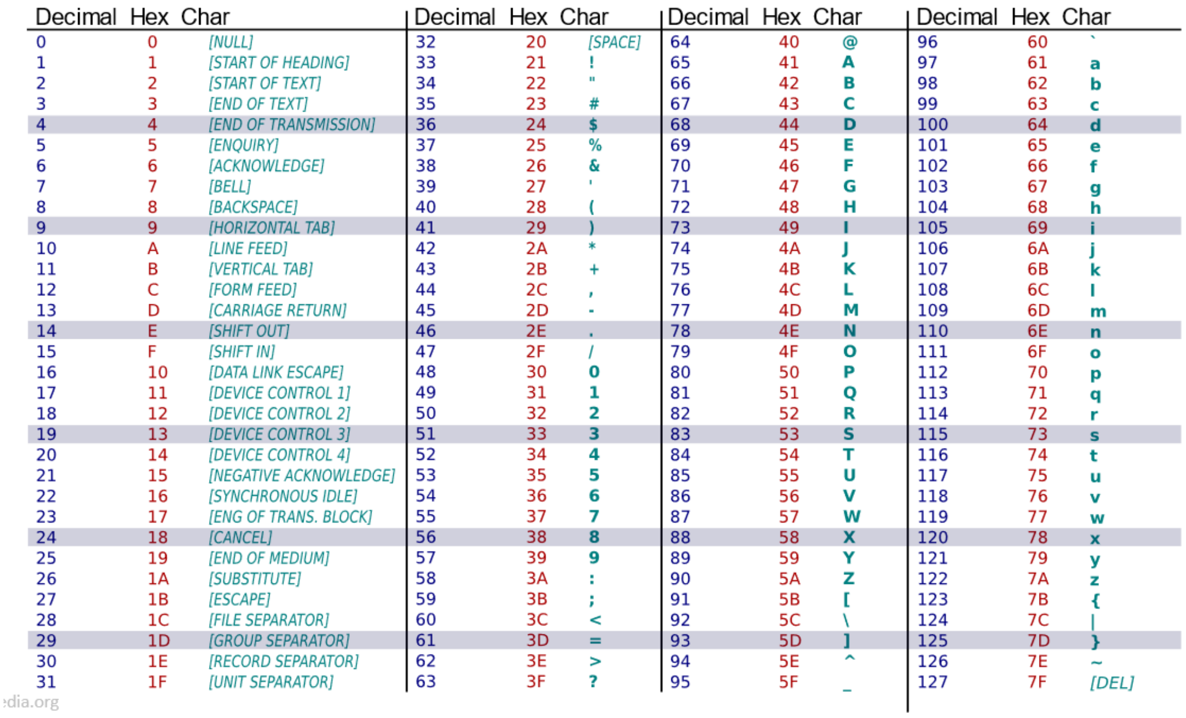
Binary: Base 2

0, 1

Hexadecimal: Base 16

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

ASCII Table



Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Encoding

It is a concept that tells you represent characters into numbers (or any other specific format).

ASCII (American Standard Code for Information Interchange)

- Each character occupies 7 bits (generally considered as 1 byte)
- 128 characters (0 to 127)
- Includes all keyboard characters with alphabets, digits, special characters etc.

UTF-16 / Unicode (Universal Code)

- Each character occupies 2 or 4 bytes (generally considered as 2 bytes)
- About 144697 characters (approx)
- Includes all natural language characters along with ASCII.

'System.IO' namespace

This namespace contains classes to perform File I/O operations.

System.IO:

1. class File
2. class DriveInfo
3. class FileStream
4. class Directory
5. class FileNotFoundException
6. class FileInfo
7. class StreamWriter
8. class BinaryWriter
9. class DirectoryInfo
10. class StreamReader
11. class BinaryReader

class File

It is a static class that manipulates file.

class Directory

It is a static class that manipulates directory (folder).

class FileInfo

It is a class that represents a file on the disk and performs manipulations on files.

class DirectoryInfo

It is a class that represents a directory (folder) on the disk and performs manipulations on directories.

class DriveInfo

It is a class that represents a drive and performs manipulations on drives.

class FileStream

It is a class that performs file I/O operations.

class StreamWriter

It is a class that writes text data into the file.

class StreamReader

It is a class that reads text data from the file.

class BinaryWriter

It is a class that writes binary data into the file.

class BinaryReader

Class that reads binary data from the file.

File

Static class that manipulates file.

All methods of this class are static methods.

Eg:

System.IO.File.Method( );

Methods of 'File'

1. static FileStream Create(string path)

Create / overwrites a file at the specified path.

2. static bool Exists(string path)

Determines whether the file exists in the disk or not.

3. static void Copy(string sourceFile, string destFile)

Copies the source file to the destination location.

4. static void Move(string sourceFile, string destFile)

Moves the source file to the destination location.

5. static void Delete (string path)

Deletes the specified file permanently.

6. static void WriteAllLines(string path, IEnumerable<string> contents)

Creates / overwrites a file; writes specified lines of content to the file; and close the file.

7. static string[ ] ReadAllLines(string path)

Reads file content as text and returns all lines.

8. static string ReadAllText(string path)

Reads file content as text and returns the same.

9. static void WriteAllText(string path, string contents)

Creates / overwrites a file; writes specified content to the file; and close the file.

10. static FileStream Open(string path, FileMode mode, FileAccess access)

Opens the file in specified file mode with specified file access permission and returns a new object of FileStream type.

11. static FileStream OpenRead(string path)

Opens the file in 'Open' mode and returns a new object of FileStream type.

12. static FileStream OpenWrite(string path)

Opens the file in 'OpenOrCreate' mode and returns an object of FileStream type.

FileInfo

Represents a file and provides methods to manipulate the file.

Eg:

FileInfo referenceVariable = new FileInfo(“Your File Path Here”);

Constructors of 'FileInfo'

1. FileInfo(string filePath)

It initializes the file path which needs to be manipulated.

Methods of 'FileInfo'

1. FileInfo CopyTo(string destFilePath, bool overwrite)

Copies the file into the new destination path.

It returns an object of FileInfo class, that represents the newly created file.

2. void MoveTo(string destFilePath)

Moves the file into the new destination path (should be in the same drive).

3. void Delete( )

Deletes the file permanently.

4. FileStream Create( )

Creates the file at specified path & creates and returns an object of FileStream class with 'Create' mode, which can write data to the same file.

5. FileStream Open(FileMode mode, FileAccess access)

Opens the file at specified file mode and specified file access permission & creates and returns an object of FileStream class that can write / read the file data.

6. FileStream OpenRead( )

Opens the file in 'Read' mode & creates and returns an object of FileStream class that can read the file data.

7. FileStream OpenWrite( )

Opens the file in 'OpenOrCreate' mode & creates and returns an object of FileStream class that can read the file data.

8. StreamWriter CreateText( )

Opens the file in 'Create' mode & creates and returns an object of StreamWriter class that can write text to the file.

9. StreamWriter AppendText( )

Opens the file in 'Append' mode & creates and returns an object of StreamWriter class that can write the appended text to the file.

10. StreamReader OpenText( )

Opens the file in 'Open' mode & creates and returns an object of StreamReader class that can read text from the file.

Properties of 'FileInfo'

bool Exists

Determines whether the file exists in the disk or not.

string FullName

Represents full path (including file name and extension) of the file.

string Name

Represents only name of the file (without path).

string DirectoryName

Represents only path of the file (without file name).

string Extension

Represents only file extension (without file name).

DateTime CreationTime

Represents date and time of file creation.

DateTime LastWriteTime

Represents date and time of last modification of the file.

DateTime LastAccessTime

Represents date and time of last access of the file.

long Length

Represents file size (in the form of no. of bytes).

Directory

Static class that manipulates directory (folder).

All methods of this class are static methods.

Eg: System.IO.Directory.Method( );

Methods of 'Directory'

1. static DirectoryInfo CreateDirectory(string path)

Create a directory at the specified path.

2. static void Delete (string path, bool recursive)

true: Deletes the directory including sub directories and all files.

false: Deletes the directory only if it is empty.

3. static bool Exists(string path)

Determines whether the directory exists in the disk or not.

4. static string[ ] GetDirectories(string path)

Returns string[ ] that contains paths of sub directories of specified directory.

5. DirectoryInfo[ ] GetDirectories (string searchPattern)

Returns DirectoryInfo[ ] that represents sub directories that matches with specified search pattern.

6. FileInfo[ ] GetFiles( )

Returns FileInfo[ ] that represents files of current directory.

7. FileInfo[ ] GetFiles(string searchPattern)

Returns FileInfo[ ] that represents files that matches with specified search pattern.

8. void MoveTo(string destDirName)

Moves the current directory to the specified location in the same

DirectoryInfo

Class that represents a directory (folder) on the disk and performs manipulations on directories.

Eg:

DirectoryInfo referenceVariable = new DirectoryInfo(“Directory Path Here”);

Constructors of 'DirectoryInfo'

1. DirectoryInfo(string path)

It initializes the directory path which needs to be manipulated.

Methods of 'DirectoryInfo'

1. void Create( )

Create the directory at the current path.

2. DirectoryInfo CreateSubDirectory (string path)

Creates a sub directory at the current path with specified name.

3. void Delete(bool recursive)

true: Deletes the directory including sub directories.

false: Deletes the directory only if it is empty.

4. DirectoryInfo[ ] GetDirectories()

Returns DirectoryInfo[ ] that represents sub directories of current directory.

5. DirectoryInfo[ ] GetDirectories (string searchPattern)

Returns DirectoryInfo[ ] that represents sub directories that matches with specified search pattern.

6. FileInfo[ ] GetFiles( )

Returns FileInfo[ ] that represents files of current directory.

7. FileInfo[ ] GetFiles(string searchPattern)

Returns FileInfo[ ] that represents files that matches with specified search pattern.

8. void MoveTo(string destDirName)

Moves the current directory to the specified location in the same drive.

Properties of 'DirectoryInfo'

1. bool Exists

Determines whether the directory exists in the disk or not.

2. string FullName

Represents full path of the directory

3. string Name

Represents only name of the directory (without path).

4. DirectoryInfo Parent

Represents parent directory of the current directory.

5. DirectoryInfo Root

Represents root (drive) of the directory.

6. DateTime CreationTime

Represents date and time of directory creation.

7. DateTime LastWriteTime

Represents date and time of last modification of the directory.

8. DateTime LastAccessTime

Represents date and time of last access of the directory.

DriveInfo

Class that represents a drive and performs manipulations on drives.

You can read both fixed / removable drives.

Eg:

DriveInfo referenceVariable = new DriveInfo(“Your Drive Name Here”);

Constructor of 'DriveInfo'

1. DriveInfo(string path)

It initializes the drive name which needs to be manipulated.

Properties of 'DriveInfo'

1. string Name

Represents name of the drive.

2. string DriveType

Represents type of drive either 'Fixed' or 'Removable'

3. string VolumeLabel

Represents label of the drive (set by user).

4. DirectoryInfo RootDirectory

Represents root directory of the drive.

5. long TotalSize

Represents total size (bytes) of the drive.

6. long AvailableFreeSpace

Represents total free space (bytes) of the drive.

FileStream

Class that performs file I/O operations.

Writes / reads data in byte[] format.

Eg:

FileStream referenceVariable = new FileStream("File Path", FileMode.Create, FileAccess.Write);

Constructors of 'FileStream'

1. FileStream(string filePath, FileMode mode, FileAccess access)

It initializes opens the file at specified mode and specified access permission.

FileMode: CreateNew, Create, Open, OpenOrCreate, Append

FileAccess: Read, Write, ReadWrite

'FileMode' in FileStream

1. CreateNew

It specifies that the o/s should create a new file.

If the file already exists an IOException will be thrown.

2. Create

It specifies that the o/s should create a new file.

If the file already exists, it will be overwritten.

3. Open

It specifies that the o/s should open an existing file.

If the file doesn't exist, a FileNotFoundException will be thrown.

4. OpenOrCreate

It specifies that the o/s should open an existing file.

If the file doesn't exist, a new file will be created.

It is useful for reading content of the file.

5. Append

It specifies that the o/s should open an existing file; and seek to end of the file, in order to write content at the end.

It is useful with FileAccess.Write

'FileAccess' in FileStream

1. Read

It is used to read content from an existing file.

2. Write

It is used to write content to a file.

3. ReadWrite

It can be used for both read / write operations in a file.

Methods of 'FileStream'

1. void Write(byte[] array, int offset, int count)

Writes the specified no. of bytes specified by count, based on the byte[] specified by 'array' into the file, after the no. of bytes specified by 'offset'.

2. int Read(byte[] array, int offset, int count)

Read the specified no. of bytes specified by count, into the byte[] specified by 'array' from the file, after the no. of bytes specified by 'offset'.

3. void Close( )

Closes the file.

StreamWriter

Class that writes text data into the file.

Internally uses FileStream.

Eg:

StreamWriter referenceVariable = new StreamWriter("File Path");

Constructors of 'StreamWriter'

1. StreamWriter(FileStream stream)

It initializes the StreamWriter based on the specified FileStream.

2. StreamWriter(string path)

It initializes the StreamWriter for the specified file path.

Methods of 'StreamWriter'

1. void Write(string content)

Writes the specified string content to the file.

2. void Close( )

Close the file.

StreamReader

Class that reads text data from the file.

Internally uses FileStream.

Eg:

StreamReader referenceVariable = new StreamReader("File Path");

Constructors of 'StreamReader'

1. StreamReader(string path)

It initializes the StreamReader for the specified file path.

2. StreamReader(FileStream stream)

It initializes the StreamReader based on the specified FileStream.

Methods of 'StreamReader'

1. int Read(char[] buffer, int startIndex, int count)

Reads the specified number of characters (using 'count') from the file into the specified buffer (char[]) and inserts the characters at specified index of buffer.

2. string ReadToEnd( )

Reads complete content of the file in text format and returns the same as a string.

3. string ReadLine( )

Reads a line from the file in text format and returns the same as a string.

4. void Close( )

Close the file.

BinaryWriter

Class that writes binary data into the file.

Internally uses FileStream.

For example, 65 is written as 100 0001.

Eg:

BinaryWriter referenceVariable = new BinaryWriter(fileStream);

Constructors of 'BinaryWriter'

1. BinaryWriter(FileStream stream)

It initializes the BinaryWriter based on the specified FileStream.

Methods of 'BinaryWriter'

1. void Write(…)

Writes any type of value (only primitive types / string) into the file.

2. void Close( )

Close the file.

BinaryReader

Class that reads binary data from the file.

Internally uses FileStream.

Eg:

BinaryReader referenceVariable = new BinaryReader(fileStream);

Methods of 'BinaryReader'

1. byte ReadByte( )

Reads and returns a byte value from the file.

2. ReadSByte( ), ReadInt16( ), ReadUInt16( ), ReadInt32( ), ReadUInt32( ), ReadInt64( ), ReadUInt64( ), ReadSingle( ), ReadDouble( ), ReadDecimal( ), ReadChar( ), ReadString( ), ReadBoolean( )

Reads and returns the specific type of value from the file.

3. string Close( )

Closes the file.

BinaryFormatter

Class that serializes (converts) an object-state into binary format into and stores in binary file.

Serialization is a process of converting an object from one format to another format.

It can also read existing object-state from the binary file.

Full Path: System.Runtime.Serialization.Formatters.Binary.BinaryFormatter

Eg:

BinaryFormatter referenceVariable = new BinaryFormatter( );

Methods of 'BinaryFormatter'

1. void Serialize(FileStream FileStream, object data)

Converts the object-state into the binary file, using the specified FileStream (with Write access).

2. object Deserialize(FileStream FileStream)

Reads the existing object-state from the binary file, using the specified FileStream (with Read access).

JavaScriptSerializer

Class that serializes (converts) an object-state into JSON format.

It can also convert JSON data into object of any class.

Full Path: System.Web.Script.Serialization.JavaScriptSerializer

Eg:

//serialization:

javaScriptSerializer.Serialize(YourObject);

//deserialization:

javaScriptSerializer.Deserialize(string jsonData, typeof(ClassName));

XmlSerializer

Class that serializes (converts) an object-state into XML format and stores in XML file.

It can also read existing object-state from the XML file.

Full Path: System.Xml.Serialization.XmlSerializer

XmlSerializer

XmlSerializer referenceVariable = new XmlSerializer(typeof(ClassName));

Methods of 'XmlSerializer'

1. void Serialize(FileStream FileStream, object data)

Converts the object-state into the xml file, using the specified FileStream (with Write access).

2. object Deserialize(FileStream fileStream)

Reads the existing object-state from the xml file, using the specified FileStream (with Read access).