C# Basics. Methodical Notes.

1. Minimum Learning Outcome

At minimum, after the learning you should be able to comprehend *at least* these C# concepts:

1. Keywords:

abstract	decimal	foreach	override	struct
as	default	if	params	switch
base	delegate	in	private	this
bool	do	int	protected	throw
break	double	interface	public	true
byte	else	internal	readonly	try
case	enum	is	ref	using
catch	event	new	return	virtual
char	false	null	sealed	void
class	finally	object	short	while
const	float	operator	static	yield
continue	for	out	string	

2. Language constructs and mechanisms:

class inheritance	garbage collection	properties
classes	generics	static classes and members
constructors	interface implementation	strings manipulation
delegates and lambdas	interfaces	structs
enums	methods overloading	type conversions
events	nullable types	virtual methods
exceptions handling	objects disposal	
extension methods	objects equality	

3. Framework constructs and mechanisms: collections, dictionaries, streams, LINQ

4. Standard means to work with: files, HTTP requests

2. Recommended Readings

The primary recommended source is a nice book ***C# 6.0 in a Nutshell*** **by Joseph and Ben Albahari**. This is a perfectly self-contained book containing all you may need to quickly grasp all aspects of C# and .NET for the beginner level and even further! **Code examples** for this book are available on book site http://www.albahari.com/nutshell/code.aspx

As a supporting material, please, check MSDN https://msdn.microsoft.com. MSDN has lots of walkthroughs, guides, as well as official C#/.NET documentation and more. Combined with a book by Albahari, it gives a perfect learning bunch! You hardly need anything else for now.

Still, for the most curious among you, we highly recommend some cool advanced literature with an in-depth view on C#/.NET internals and tricks: ***C# In Depth*** **by Jon Skeet** and ***CLR via C#*** **by Jeffrey Richter**.

Note: all mentioned books are available in the course on Distedu.

3. Self-control Questions

First, you should be able to explain meaning and usage of keywords and concepts mentioned in 1.1 and 1.2. We don't expect you to know it all in depth, of course! However, at least you should be aware of them and know where/how/why each concept is used.

Additionally, here are some questions that will help you to check your theoretical understanding of some critical C# principles:

1. What is the difference between classes and structs?
2. What are value types?
3. What are reference types?
4. What is heap memory?
5. What is stack memory?
6. What method parameters are passed by value?
7. What method parameters are passed by reference?
8. What size (in bytes) is the variable of type class Student?
9. What does *null* mean?
10. What is the difference between *for* and *foreach*?
11. What is the difference between *const* and *readonly*?
12. What is the difference between *float* and *decimal*? When to use each type?
13. What is the default access modifier for class members?
14. What is the purpose of static fields?
15. How to prevent a class from being inherited?
16. How to prevent a method from being overridden?
17. Can struct inherit from other structs?
18. Can struct implement an interface?
19. Can class inherit from a struct?
20. Can struct be null?
21. Is it possible to have an *int* with no value (like null)?
22. Can an abstract class have methods implementations?
23. Can an interface have methods implementations?
24. Can an interface have fields?
25. Can a derived class skip implementation of an abstract method?
26. Can a class skip implementation of an interface method?
27. Can *int* be directly assigned to *object*?
28. Can *object* be directly assigned to *int*?
29. Can *object* be casted to *int*?
30. What is boxing?
31. What is the difference between IEnumerable and IList?
32. What is the difference between List and array?
33. What is the difference between IDictionary (Hashtable) and IDictionary<TKey, TValue>?
34. What is the algorithmic complexity of array operations (add, remove, access by index)?
35. What is the algorithmic complexity of List operations (add, remove, access by index)?
36. What is the algorithmic complexity of accessing by key an element in a Hashtable?
37. What is the algorithmic complexity of LinkedList operations (add, remove, access by index)?

Important: Don't just find answers to these questions! Instead read through the corresponding topics carefully, and then use questions to check your understanding.

4. Methodical Guidelines and Recommendations

Here are some suggestions on how to study all necessary material easily and with as much fun as possible ☺

Option #1

For those of you, who prefer pure reading, you can simply go through Albahari's book. You may choose your own reading method, but one possible approach is reading it iteratively several times, grasping more and more ideas and concepts with each reading. Albahari's book is pretty self-contained, with examples and necessary level of details.

Option #2

Some of you may prefer not to read a single book, but study C# concept by concept, playing with examples, separate sources etc. This also can be a good approach, if you like it. Just use keywords and concepts from 1.1 and 1.2 to guide yourself through the language basics. But make sure you understand not only the syntax, but the semantics as well!

In General

You should work towards a broad understanding rather than a deep one. It is very important to get a full picture of (the subset of) C# and be able to solve tasks and comprehend existing code.