C# Programming

What is C#

C# is a general-purpose, modern and object-oriented programming language pronounced as "C Sharp". It was developed by Microsoft led by Anders Hejlsberg and his team within the *.NET* initiative and was approved by the European Computer Manufacturers Association (ECMA) and International Standards Organization (ISO). C# is among the languages for Common Language Infrastructure. C# is a lot similar to Java syntactically and is easy for users who have knowledge of C, C++ or Java.

C# | Identifiers

In programming languages, identifiers are used for identification purposes. Or in other words, identifiers are the user-defined name of the program components. In C#, an identifier can be a class name, method name, variable name, or label.

Example:

```
public class GFG {
   static public void Main ()
   {
        int x;
   }
}
```

Here the total number of identifiers present in the above example is 3 and the names of these identifiers are:

- **GFG:** Name of the class
- **Main:** Method name
- **x:** Variable name

Rules for defining identifiers in C#:

There are certain valid rules for defining a valid C# identifier. These rules should be followed, otherwise, we will get a compile-time error.

- The only allowed characters for identifiers are all alphanumeric characters(**[A-Z]**, **[a-z]**, **[0-9]**), '_' (underscore). For example "geek@" is not a valid C# identifier as it contain '@' – special character.
- Identifiers should not start with digits([0-9]). For example "123geeks" is not valid in the C# identifier.

- Identifiers should not contain white spaces.

- Identifiers are not allowed to use as keywords unless they include @ as a prefix. For example, **@as** is a valid identifier, but “**as**” is not because it is a keyword.
- C# identifiers allow Unicode Characters.
- C# identifiers are case-sensitive.
- C# identifiers cannot contain more than 512 characters.
- Identifiers do not contain two consecutive underscores in their name because such types of identifiers are used for the implementation.

C# | Keywords

Keywords or Reserved words are the words in a language that are used for some internal process or represent some predefined actions. These words are therefore not allowed to use as variable names or objects. Doing this will result in a **compile-time error**.
Example:

```
// C# Program to illustrate the keywords

using System;



class GFG {



    // Here static, public, void

    // are keywords

    static public void Main () {



        // here int is keyword

        // a is identifier
        int a = 10;



        Console.WriteLine("The value of a is: {0}",a);



        // this is not a valid identifier



        // removing comment will give compile time error

        // double int = 10;



    }

}
```

Output:

```
The value of a is: 10
```

There are total 78 keywords in C# as follows:

abstract	do	in	protected	throw
as	double	int	public	true
base	else	interface	readonly	try
bool	enum	internal	ref	typeof
break	event	is	return	unit
byte	explicit	lock	sbyte	ulong
case	extern	long	sealed	unchecked
catch	false	namespace	short	unsafe
char	finally	new	sizeof	ushort
checked	fixed	null	stackalloc	using
class	float	object	static	using static
const	for	operator	string	virtual
continue	foreach	out	struct	void
decimal	goto	override	switch	volatile
default	if	params	this	while
delegate	implicit	private		

Keywords in C# is mainly divided into 10 categories as follows:

1. **Value Type Keywords:** There are **15** keywords in value types which are used to define various data types.

bool	byte	char	decimal
double	enum	float	int
long	sbyte	short	struct
unit	ulong	ushort	

Example:

```
// C# Program to illustrate the

// value type keywords

using System;



class GFG {



    // Here static, public, void

    // are keywords

    static public void Main () {



        // here byte is keyword

        // a is identifier

        byte a = 47;

        Console.WriteLine("The value of a is: {0}",a);





        // here bool is keyword

        // b is identifier

        // true is a keyword
```

```
            bool b = true;



            Console.WriteLine("The value of b is: {0}",b);

        }

}
```

Output:

```
The value of a is: 47

The value of b is: True
```

2. **Reference Type Keywords:** There are **6** keywords in reference types which are used to store references of the data or objects. The keywords in this category are: ***class, delegate, interface, object, string, void***.

3. **Modifiers Keywords:** There are **17** keywords in modifiers which are used to modify the declarations of type member.

public	private	internal	protected	abstract
const	event	extern	new	override
partial	readonly	sealed	static	unsafe
virtual	volatile			

Example:

```
// C# Program to illustrate the

// modifiers keywords

using System;



class Geeks {



    class Mod

    {



        // using public modifier

        // keyword

        public int n1;



    }



    // Main Method

    static void Main(string[] args) {



        Mod obj1 = new Mod();
        // access to public members
        obj1.n1 = 77;


        Console.WriteLine("Value of n1: {0}", obj1.n1);
      }
  }
```

Output:

```
Value of n1: 77
```

4. **Statements Keywords:** There are total **18** keywords which are used in program instructions.

if	else	switch	do	for
foreach	in	while	break	continue
goto	return	throw	try	catch
finally	checked	unchecked		

Example:

```
// C# program to illustrate the statement keywords

using System;



class demoContinue

{

    public static void Main()

    {



        // using for as statement keyword

        // GeeksforGeeks is printed only 2 times

        // because of continue statement

        for(int i = 1; i < 3; i++)

        {



            // here if and continue are keywords

            if(i == 2)

            continue;



            Console.WriteLine("GeeksforGeeks");
        }

    }

}
```

Output:
```
GeeksforGeeks
```

5. **Method Parameters Keywords:** There are total **4** keywords which are used to change the behavior of the parameters that passed to a method. The keyword includes in this category are: ***params, in, ref, out***.

6. **Namespace Keywords:** There are total **3** keywords in this category which are used in namespaces. The keywords are: ***namespace, using, extern***.

7. **Operator Keywords:** There are total **8** keywords which are used for different purposes like creating objects, getting a size of object etc. The keywords are: ***as, is, new, sizeof, typeof, true, false, stackalloc***.

8. **Conversion Keywords:** There are **3** keywords which are used in type conversions. The keywords are: ***explicit, implicit, operator***.

9. **Access Keywords:** There are **2** keywords which are used in accessing and referencing the class or instance of the class. The keywords are ***base, this***.

10. **Literal Keywords:** There are **2** keywords which are used as literal or constant. The keywords are ***null, default***.

Important Points:
- Keywords are not used as an identifier or name of a class, variable, etc.
- If you want to use a keyword as an identifier then you must use @ as a prefix. For example, *@abstract* is valid identifier but not *abstract* because it is a keyword.

Example:
```
int a = 10;              // Here int is a valid keyword

double int = 10.67;     // invalid because int is a keyword

double @int = 10.67;   // valid identifier, prefixed with @

int @null = 0;       // valid
```

```
// C# Program to illustrate the use of

// prefixing @ in keywords

using System;



class GFG {



    // Here static, public, void

    // are keywords

    static public void Main () {



        // here int is keyword

        // a is identifier

        int a = 10;



        Console.WriteLine("The value of a is: {0}",a);

          // prefix @ in keyword int which

        // makes it a valid identifier

        int @int = 11;

                  Console.WriteLine("The value of a is: {0}",@int);
```

```
                }

}
```

Output:

```
The value of a is: 10

The value of a is: 11
```

Contextual Keywords

These are used to give a specific meaning in the program. Whenever a new keyword comes in C#, it is added to the contextual keywords, not in the keyword category. This helps to avoid the crashing of programs which are written in earlier versions.

Important Points:

- These are not reserved words.
- It can be used as identifiers outside the context that's why it named contextual keywords.
- These can have different meanings in two or more contexts.
- There are total **30** contextual keywords in C#.

add	equals	nameof	value
alias	from	on	var
ascending	get	orderby	when
async	global	partial(type)	where
await	group	partial(method)	where
by	into	remove	yield
descending	join	select	
dynamic	let	set	

Example:

```
// C# program to illustrate contextual keywords

using System;
```

```
public class Student {


    // Declare name field

    private string name = "GeeksforGeeks";


    // Declare name property

    public string Name

    {


    // get is contextual keyword

    get

        {

            return name;

        }


        // set is a contextual

        // keyword

        set

        {
            name = value;

        }

    }

}



class TestStudent {



    // Main Method

    public static void Main(string[] args)

    {

        Student s = new Student();



        // calls set accessor of the property Name,

        // and pass "GFG" as value of the

        // standard field 'value'.

        s.Name = "GFG";



        // displays GFG, Calls the get accessor

        // of the property Name.

        Console.WriteLine("Name: " + s.Name);
        // using get and set as identifier

        int get = 50;

        int set = 70;



        Console.WriteLine("Value of get is: {0}",get);

        Console.WriteLine("Value of set is: {0}",set);

    }

}
```

Output:

```
Name: GFG

Value of get is: 50

Value of set is: 70
```