Code review:

1. Namespace declaration is missing
2. Variable names are not clear. For example, 'f' could be named as 'format' and 'str' as 'value'
3. Spacing and formatting do not follow best practices. Example: { "number",true}. It is better either to include spaces before and after ('number', true } or without spaces at all ('number', true}. Same for other elements in the array
4. Variable name 'isNotAllowed' is not clear. It is common to use such variables for boolean values and not for integers.
5. It is common to start a loop with 0 value and not 1. And in this case it causes the error because the loop won't go to the last item {'timespan', true} since 3 < 3 is false.
6. Variable names allowedDict and isNotAllowed show the opposite. For clear readability it is better to use variable names with the same meaning like allowedDict and isAllowed.
7. The condition inside the loop is not providing the correct value. If the condition is true, then a variable isNotAllowed will increase which will cause the Exception, but in case if the format is allowed (condition is true), we don't need an exception but return a value.
8. Exceptions should be created as a separated public class for each type of exception, for example, UnknownFormatException or FormatNotAllowedException.
9. For both date and timespan formats the code has a DateTime.TryParse method. We can use Timespan.TryParse for timespan.
10. When the format is timespan and the user provides a number as a string ('123'), the code will recognize it as 123 days and will consider it as a correct value. Do we need this behavior?
11. If the user will provide a timespan as value for date format, DateTime.TryParse will correctly work with this, which means that value for timespan will not cause any problem if it will be provided together with date format.
12. We can use enumeration for format values like this public enum Format
13. {
14. Number,
15. Date,
16. Timespan
17. }
18. which will allow us to avoid using a loop at all and we can use allowedDict.ContainsKey(format) instead.

Please find the code which can be used instead:

```
public enum Format
    {
        Number,
        Date,
        Timespan
    }
```

```
public class Class1
    {
        public static bool IsFormat(string value, Format format)
        {
            if(!Enum.IsDefined(typeof(Format), format ) )
            {
                throw new UnknownFormatException("Unknown
format.");
            }
            var allowedDict = new Dictionary<Format, bool>()
            {
                { Format.Number,  true },
                { Format.Date, true },
                { Format.Timespan ,true }
            };
            bool isFormatAllowed =
!allowedDict.ContainsKey(format);

            if (isFormatAllowed)
                throw new FormatNotAllowedException("Format not
allowed.");
            if (format == Format.Number)
                return Int32.TryParse(value, out var _);
            else if (format == Format.Date)
                return DateTime.TryParse(value, out var _);
            else if (format == Format.Timespan)
                return TimeSpan.TryParse(value, out var _);
            else
                return false;
        }
    }
public class UnknownFormatException : Exception
    {
        public UnknownFormatException(string message, Exception
exception): base(message, exception) { }
        public UnknownFormatException(string message) :
base(message) { }
        public UnknownFormatException() : base() { }
    }

public class FormatNotAllowedException : Exception
    {
        public FormatNotAllowedException(string message, Exception
exception) : base(message, exception) { }
        public FormatNotAllowedException(string message) :
base(message) { }
        public FormatNotAllowedException() : base() { }
    }
```

```
}
```