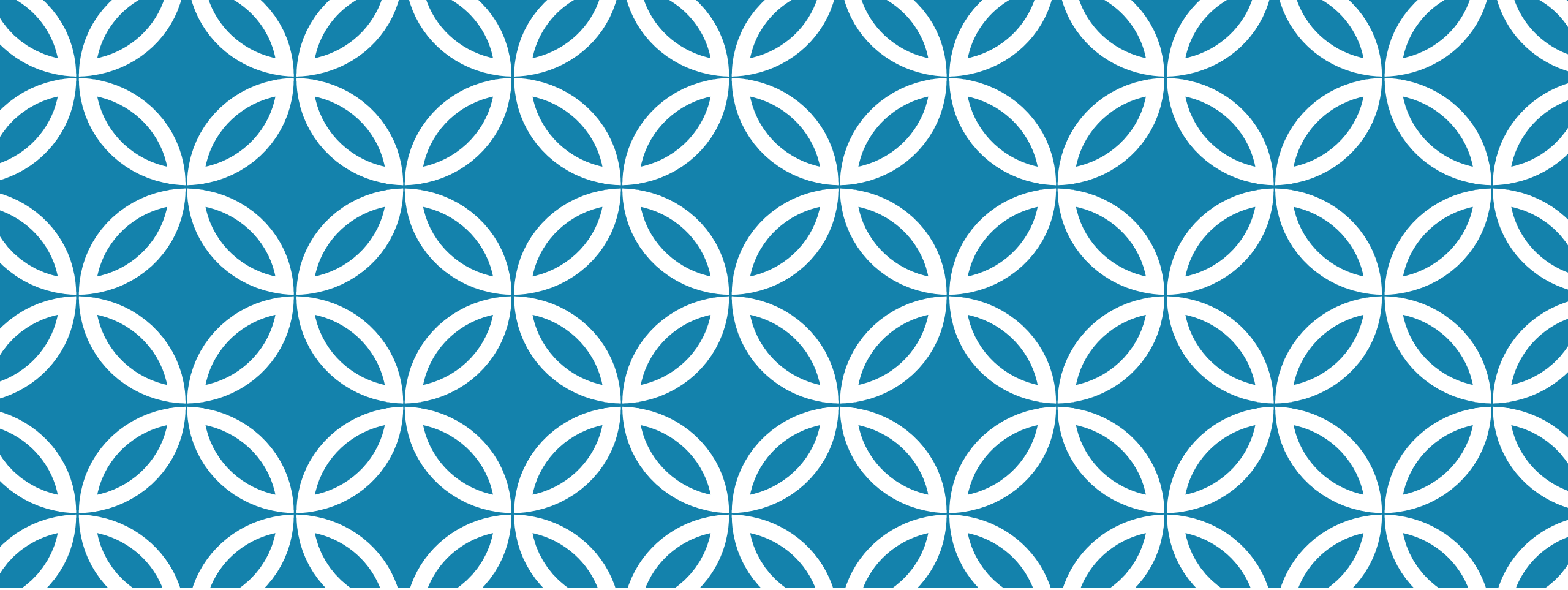
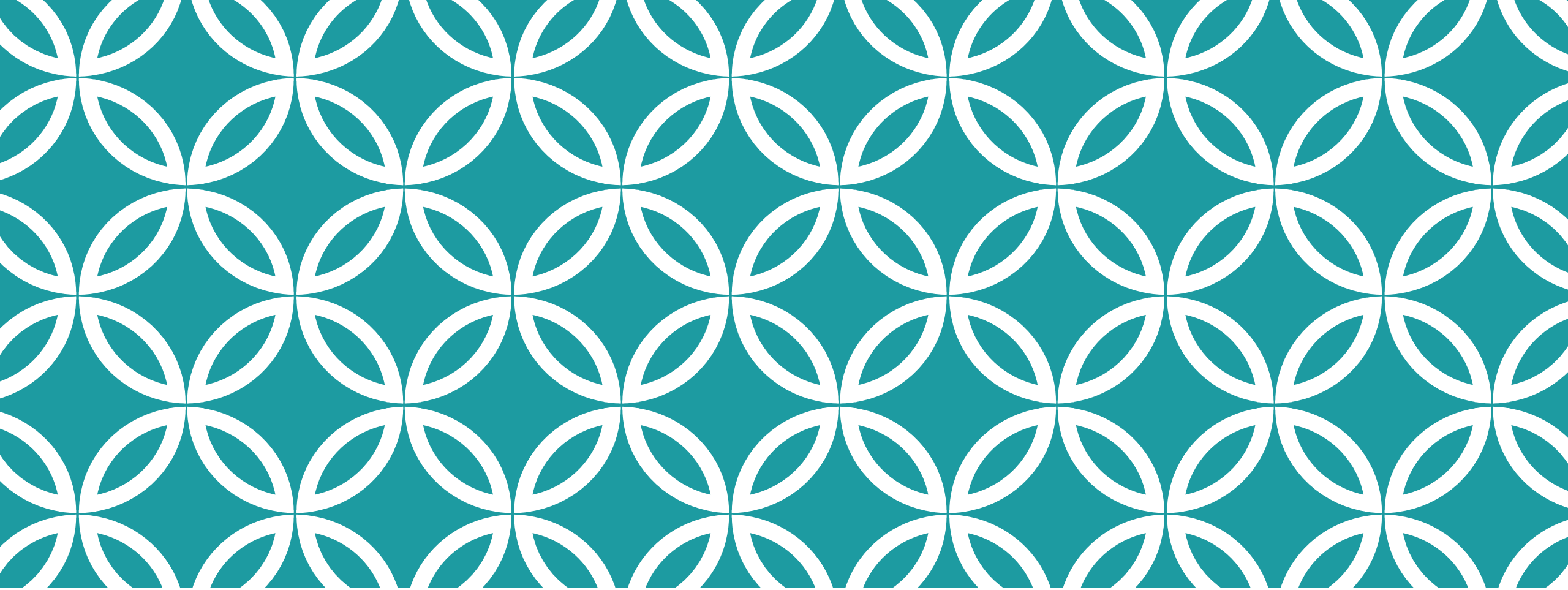


PREPROCESSOR DIRECTIVES
REGULAR EXPRESSIONS
DEBUGGING I VISUAL STUDIO
EKSEMPEL PÅ PROGRAM DER BENYTTER LINQ

Grundlæggende programmering
Lektion 12



PREPROCESSOR DIRECTIVES

Ting kompileren skal gøre inden den begynder

PREPROCESSOR DIRECTIVES

Præprocessordirektiverne giver instruktion til compileren om behandle oplysningerne, før den faktiske kompilering starter.

Alle præprocessor-direktiver begynder med #, og kun mellemrums tegn kan vises før et præprocessor-direktiv på en linje. Præprocessordirektiver er ikke udsagn, så de slutter ikke med et semikolon (;).

C# compileren har ikke en separat præprocessor; Direktiverne behandles dog som om der var en.

I C # bruges præprocessor-direktiverne til at hjælpe med betinget kompilering.
I modsætning til C og C++ direktiver bruges de ikke til at oprette makroer.

Et præprocessor-direktiv skal være den eneste instruktion på en linje.

PREPROCESSOR DIRECTIVES

Preprocessor Directive	Description.
#define	It defines a sequence of characters, called symbol.
#undef	It allows you to undefine a symbol.
#if	It allows testing a symbol or symbols to see if they evaluate to true.
#else	It allows to create a compound conditional directive, along with #if.
#elif	It allows creating a compound conditional directive.
#endif	Specifies the end of a conditional directive.
#line	It lets you modify the compiler's line number and (optionally) the file name output for errors and warnings.
#error	It allows generating an error from a specific location in your code.
#warning	It allows generating a level one warning from a specific location in your code.
#region	It lets you specify a block of code that you can expand or collapse when using the outlining feature of the Visual Studio Code Editor.
#endregion	It marks the end of a #region block.

PREPROCESSOR DIRECTIVES #DEFINE

#define preprocessor direktivet skaber symbolske konstanter.

#define lader dig definere et symbol sådan, at ved at bruge symbolet som udtrykket overført til #if-direktivet, evalueres udtrykket til ægte. Dens syntaks er:

```
#define symbol
```

Et lille program kan ses på følgende slide.

PREPROCESSOR DIRECTIVES #DEFINE

```
1   #define PI
2   using System;
3
4   namespace PreprocessorDAppl
5   {
        0 references
6       class Program
7       {
            0 references
8           static void Main(string[] args)
9           {
10  #if (PI)
11              Console.WriteLine("PI is defined");
12  #else
13              Console.WriteLine("PI is not defined");
14  #endif
15              Console.ReadKey();
16          }
17      }
18  }
```

PREPROCESSOR DIRECTIVES CONDITIONAL DIRECTIVES

Du kan bruge #if-direktivet til at oprette et betinget direktiv. Betingede direktiver er nyttige til at teste et symbol eller symboler for at kontrollere, om de er sat til sandt. Hvis de vurderer til sandt, evaluerer kompilatoren hele koden mellem #if og det næste direktiv.

Syntaks for betingede direktiver er:

```
#if symbol [operator symbol]...
```

Symbol er navnet på det symbol, du vil teste. Du kan også bruge true og false eller prepend symbolet med negation operatoren.

PREPROCESSOR DIRECTIVES CONDITIONAL DIRECTIVES

Operatørsymbolet er den operatør, der bruges til at evaluere symbolet. Operatører kan være en af følgende:

- == (equality)
- != (inequality)
- && (and)
- || (or)

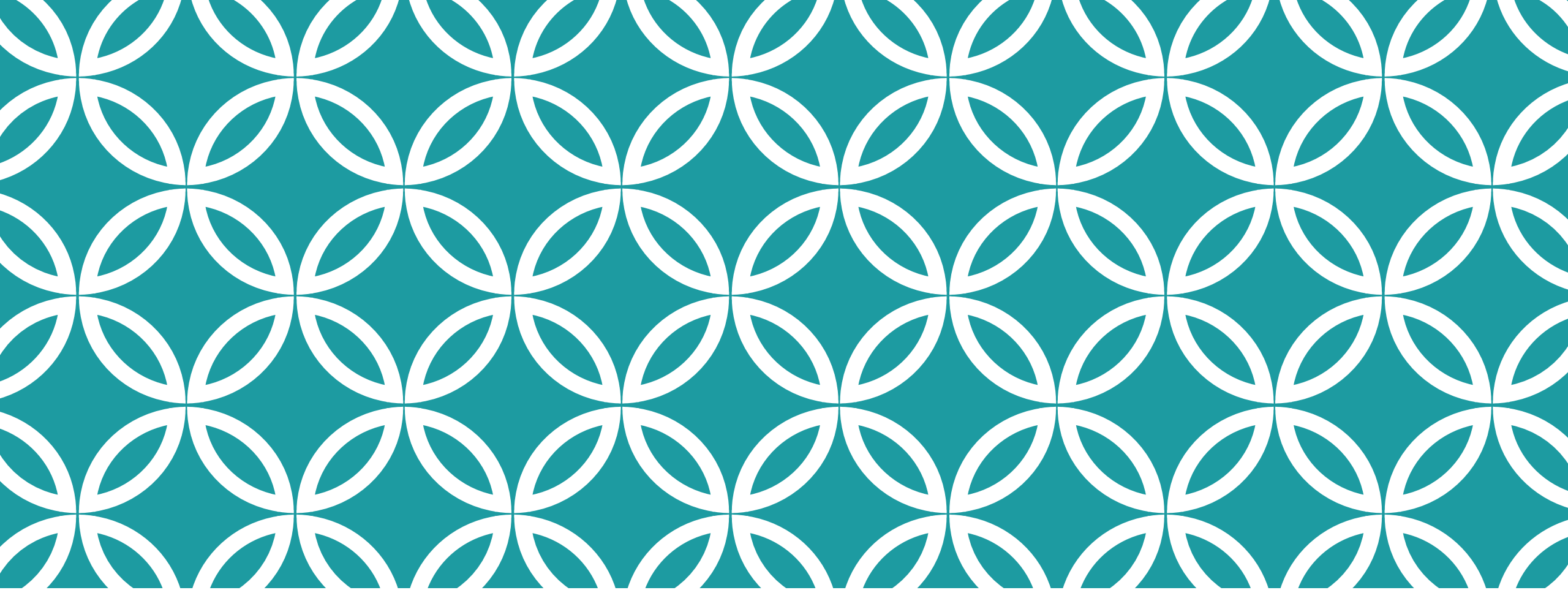
Du kan også gruppere symboler og operatører med parenteser.

Betingede direktiver bruges til at kompilere kode til et debug build eller når du kompilerer til en bestemt konfiguration. Et betinget direktiv, der begynder med et #if-direktiv, skal udtrykkeligt opsiges med et #endif-direktiv.

Et lille program kan ses på følgende slide.

PREPROCESSOR DIRECTIVES CONDITIONAL DIRECTIVES

```
1   #define DEBUG
2   #define VC_V10
3   using System;
        0 references
4   public class TestClass
5   {
            0 references
6       public static void Main()
7       {
8   #if (DEBUG && !VC_V10)
9               Console.WriteLine("DEBUG is defined");
10  #elif (!DEBUG && VC_V10)
11              Console.WriteLine("VC_V10 is defined");
12  #elif (DEBUG && VC_V10)
13              Console.WriteLine("DEBUG and VC_V10 are defined");
14  #else
15              Console.WriteLine("DEBUG and VC_V10 are not defined");
16  #endif
17              Console.ReadKey();
18      }
19  }
```



REGULAR EXPRESSIONS

Sammenligning med input

REGULAR EXPRESSIONS

Et regulært udtryk er et mønster, der kan matches imod en indtastet tekst.

.Net-frameworket kommer med en regular expression engine, der tillader sådan matching.

Et mønster består af en eller flere tegn, operatører eller konstruktioner.

Der er forskellige kategorier af tegn, operatører og konstruktioner, der lader dig definere regulære udtryk. Vi kommer ikke ind på disse her, men læs om dem i pensum til lektionen.

REGULAR EXPRESSIONS
REGEX KLASSEN

Regex klassen bruges til at repræsentere et regulært udtryk.

Den har følgende almindeligt anvendte metoder:

For den komplette liste over metoder og egenskaber, læs Microsoft-dokumentationen for C #.

Sr.No	Methods
1	**public bool IsMatch(string input)** Indicates whether the regular expression specified in the Regex constructor finds a match in a specified input string.
2	**public bool IsMatch(string input, int startat)** Indicates whether the regular expression specified in the Regex constructor finds a match in the specified input string, beginning at the specified starting position in the string.
3	**public static bool IsMatch(string input, string pattern)** Indicates whether the specified regular expression finds a match in the specified input string.
4	**public MatchCollection Matches(string input)** Searches the specified input string for all occurrences of a regular expression.
5	**public string Replace(string input, string replacement)** In a specified input string, replaces all strings that match a regular expression pattern with a specified replacement string.
6	**public string[] Split(string input)** Splits an input string into an array of substrings at the positions defined by a regular expression pattern specified in the Regex constructor.

REGULAR EXPRESSIONS

Find ord der begynder med S

\b = ligger i enden mellem et bogstav og et ikke bogstav
\S = hvilket som helst ikke mellemrums-tegn
* = matcher det tidligere element nul eller flere gange

```
1   using System;
2   using System.Text.RegularExpressions;
3
4   namespace RegExApplication
5   {
        0 references
6       class Program
7       {
            1 reference
8           private static void showMatch(string text, string expr)
9           {
10              Console.WriteLine("The Expression: " + expr);
11              MatchCollection mc = Regex.Matches(text, expr);
12              foreach (Match m in mc)
13              {
14                  Console.WriteLine(m);
15              }
16          }
17
            0 references
18          static void Main(string[] args)
19          {
20              string str = "A Thousand Splendid Suns";
21
22              Console.WriteLine("Matching words that start with 'S': ");
23              showMatch(str, @"\bS\S*");
24              Console.ReadKey();
25          }
26      }
27  }
```

REGULAR EXPRESSI

Find ord der begynder med m og slutter med e

\b = ligger i enden mellem et bogstav og et ikke bogstav
\S = hvilket som helst ikke mellemrums-tegn
* = matcher det tidligere element nul eller flere gange

```
1  using System;
2  using System.Text.RegularExpressions;
3
4  namespace RegExApplication
5  {
       0 references
6      class Program
7      {
           1 reference
8          private static void showMatch(string text, string expr)
9          {
10             Console.WriteLine("The Expression: " + expr);
11             MatchCollection mc = Regex.Matches(text, expr);
12             foreach (Match m in mc)
13             {
14                 Console.WriteLine(m);
15             }
16         }
           0 references
17         static void Main(string[] args)
18         {
19             string str = "make maze and manage to measure it";
20
21             Console.WriteLine("Matching words start with 'm' and ends with 'e':");
22             showMatch(str, @"\bm\S*e\b");
23             Console.ReadKey();
24         }
25     }
26 }
```

REGULAR EXPRESSIONS

Fjerner ekstra mellemrum

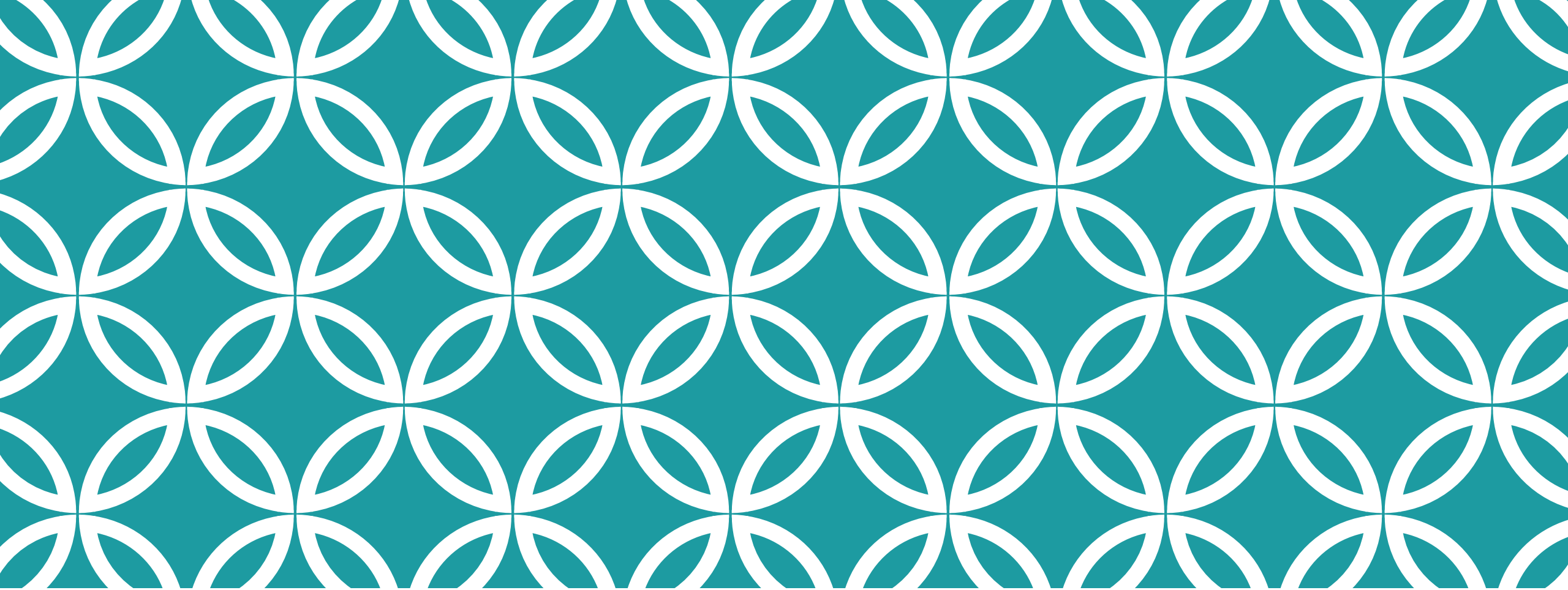
\ = når den ikke er efterfulgt af en escaped character er den det samme
\s = mellemrum
+ = matcher det tidligere element endnu en gang

MSDN

```
using System;
using System.Text.RegularExpressions;

namespace RegExApplication
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            string input = "Hello   World   ";
            string pattern = "\\s+";
            string replacement = " ";
            Regex rgx = new Regex(pattern);
            string result = rgx.Replace(input, replacement);

            Console.WriteLine("Original String: {0}", input);
            Console.WriteLine("Replacement String: {0}", result);
            Console.ReadKey();
        }
    }
}
```



DEBUGGING I VISUAL STUDIO

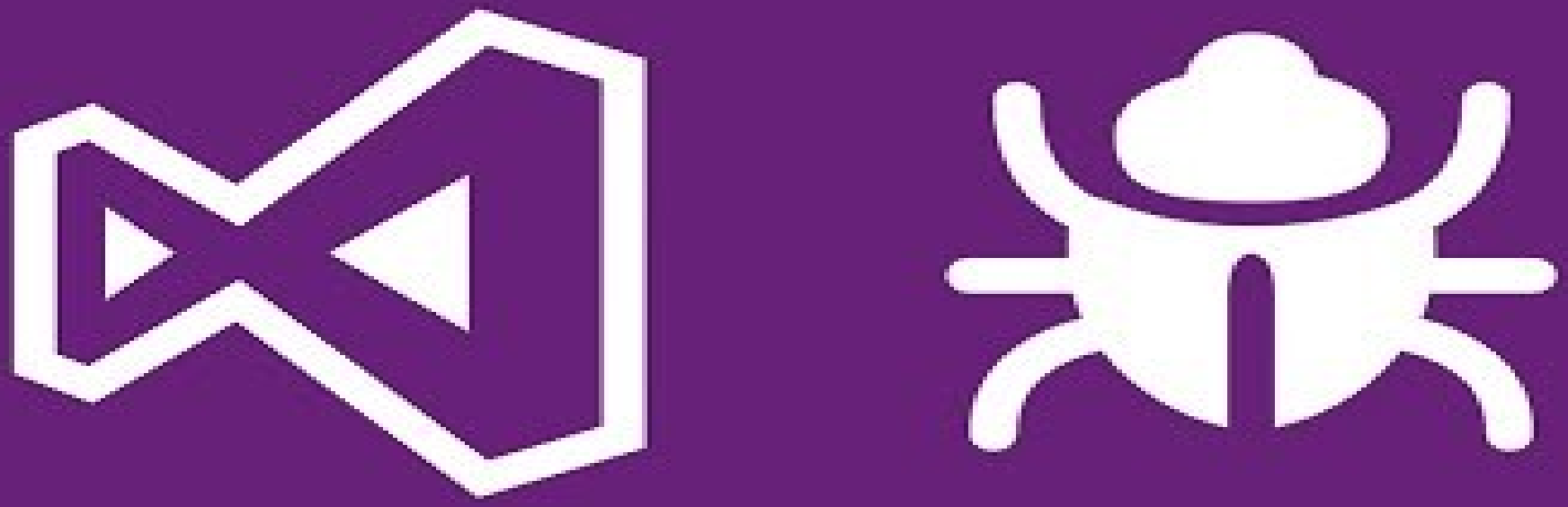
At tjekke ens program igennem

DEBUGGING I VS

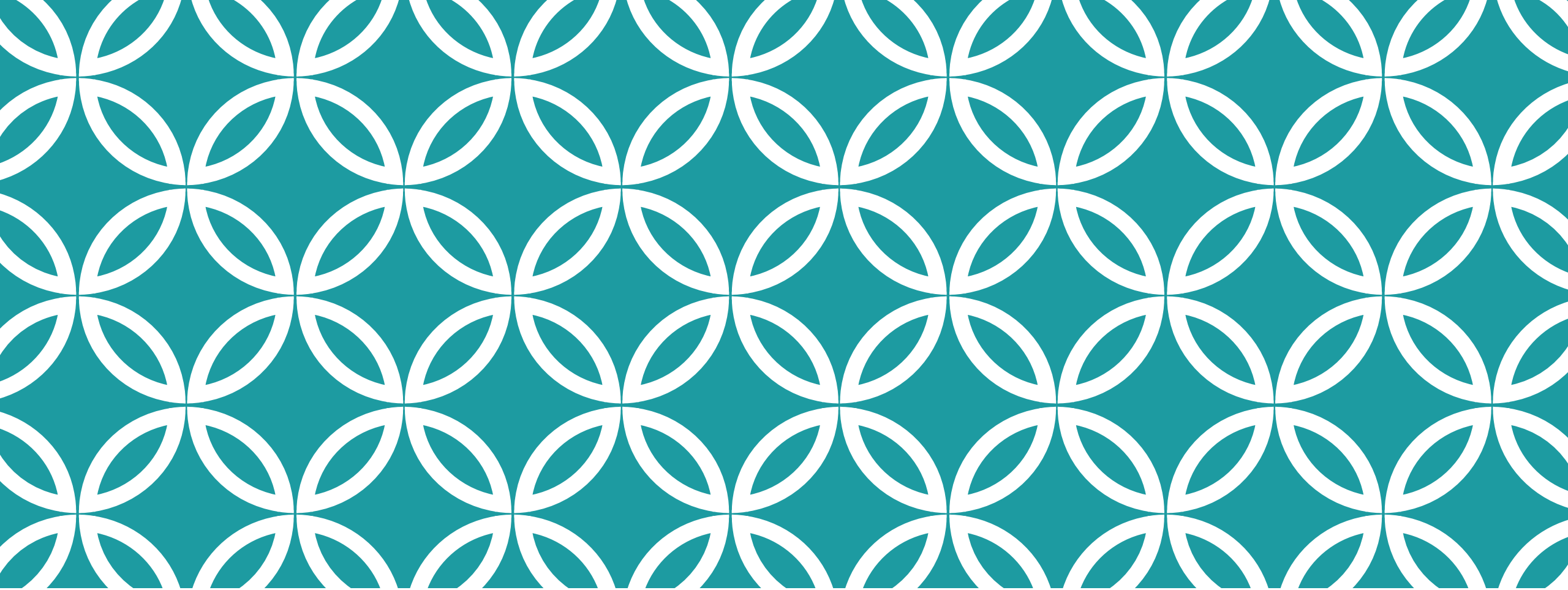
Visual Studio debugger hjælper en med at observere run-time opførsel af ens program og finde problemer.

Debuggeren fungerer med alle Visual Studio programmeringssprog og deres tilknyttede biblioteker.

Med debugger kan man afbryde udførelsen af ens program for at undersøge ens kode, undersøge og redigere variabler, se registre, se instruktioner oprettet fra ens kildekode, og se hukommelsesplads brugt af ens program.



Visual Studio 2015



EKSEMPEL PÅ PROGRAM DER BENYTTER LINQ

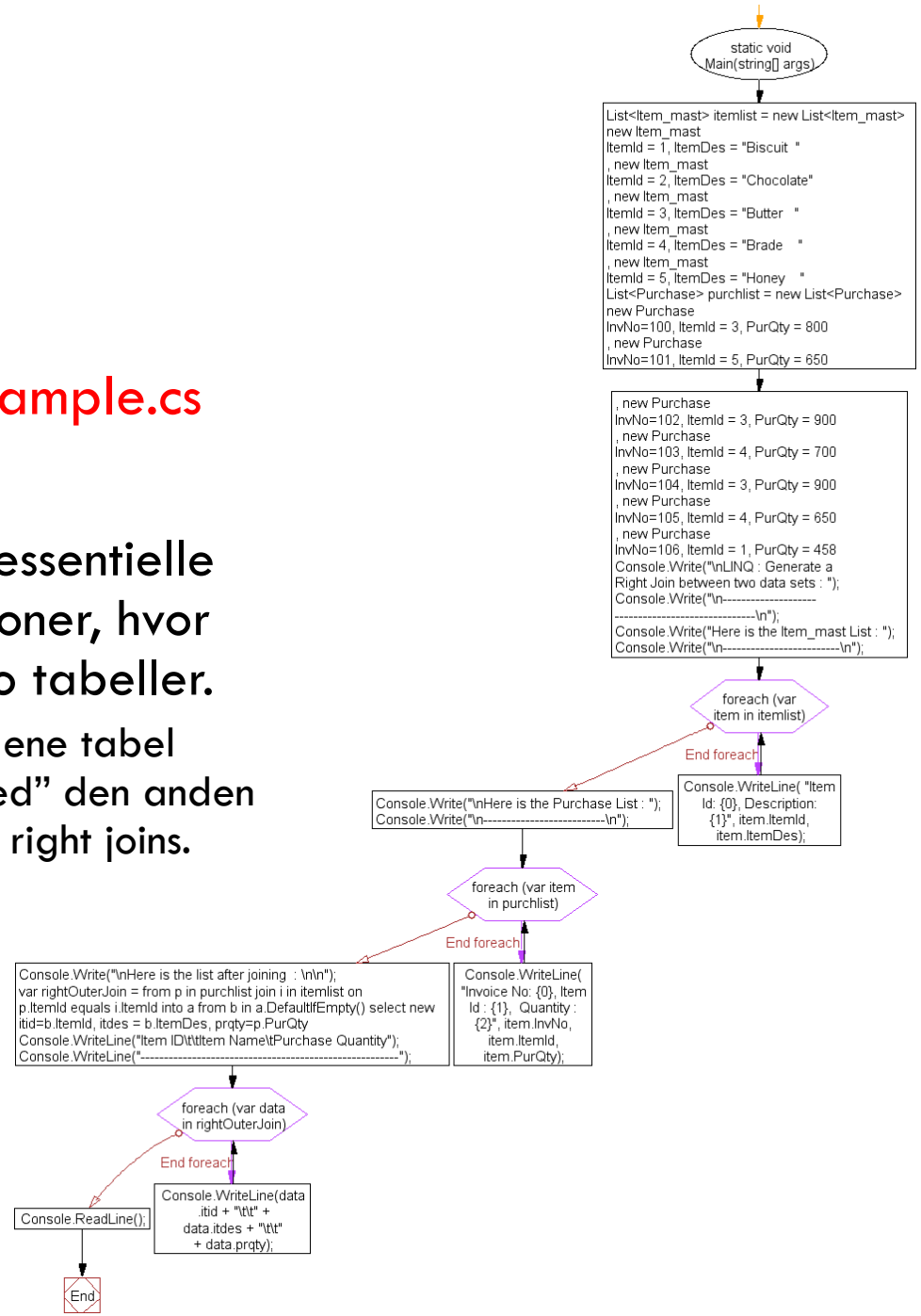
Database-operationer sammen med C#

EKSEMPEL PROGRAM

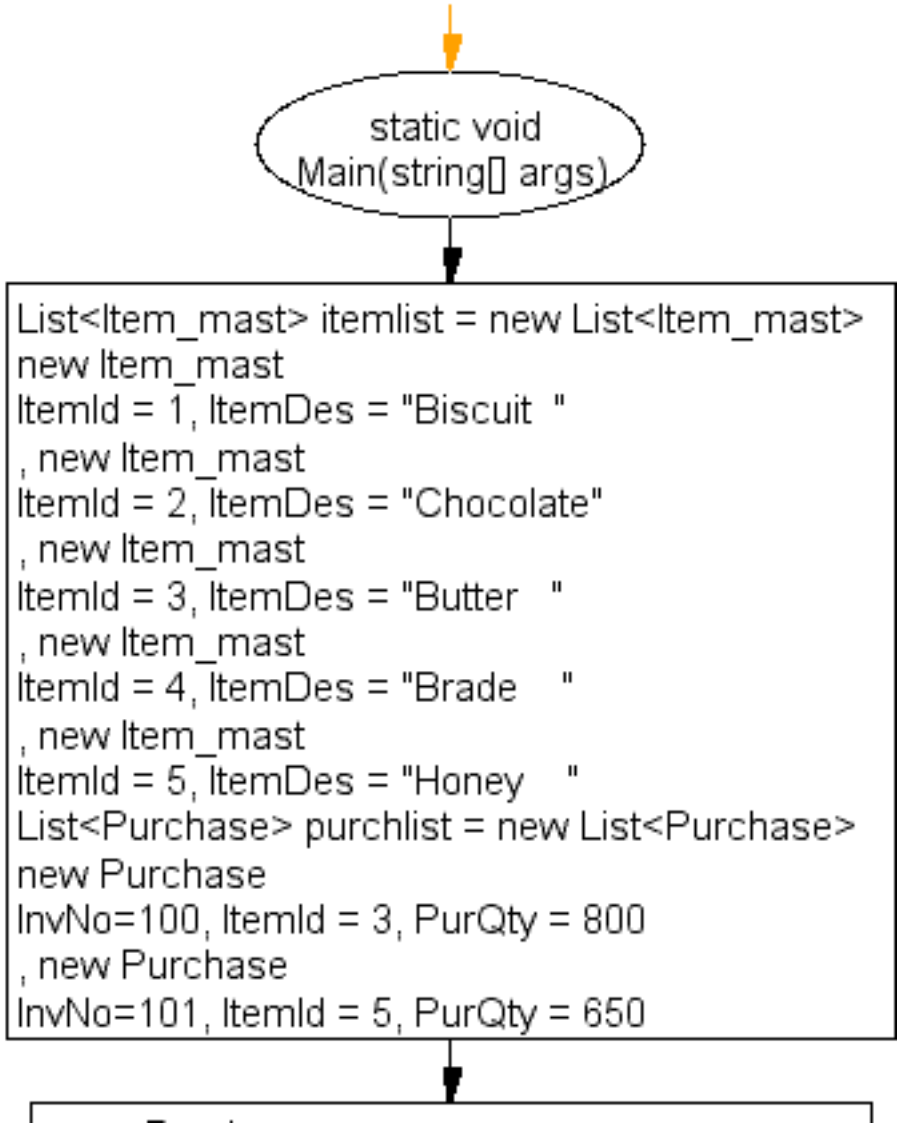
Hent lesson12example.cs fra Fronter

Joins er en af de essentielle database operationer, hvor man kombinerer to tabeller.

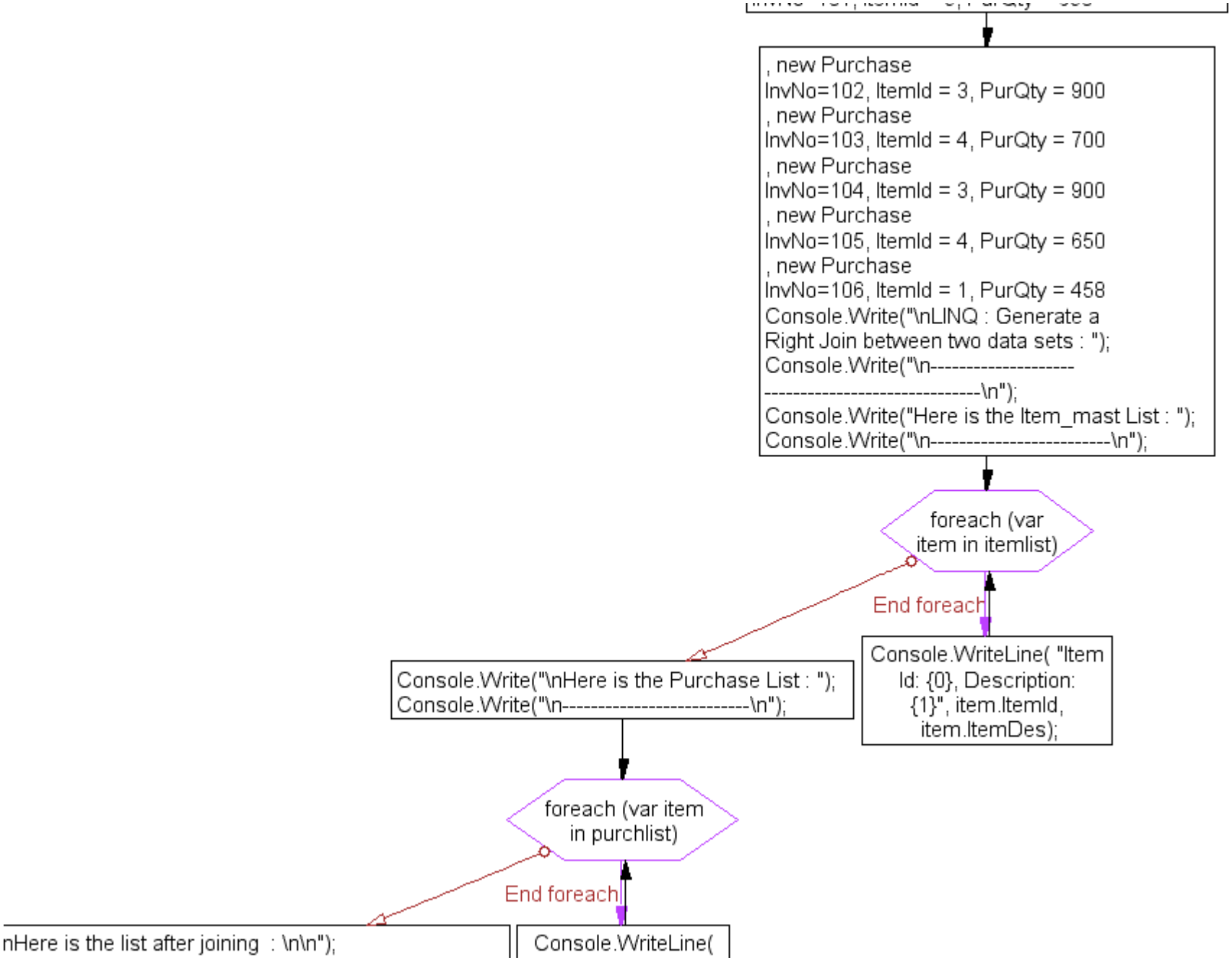
- Om man putter den ene tabel "foran" eller "bagved" den anden resulterer i left eller right joins.



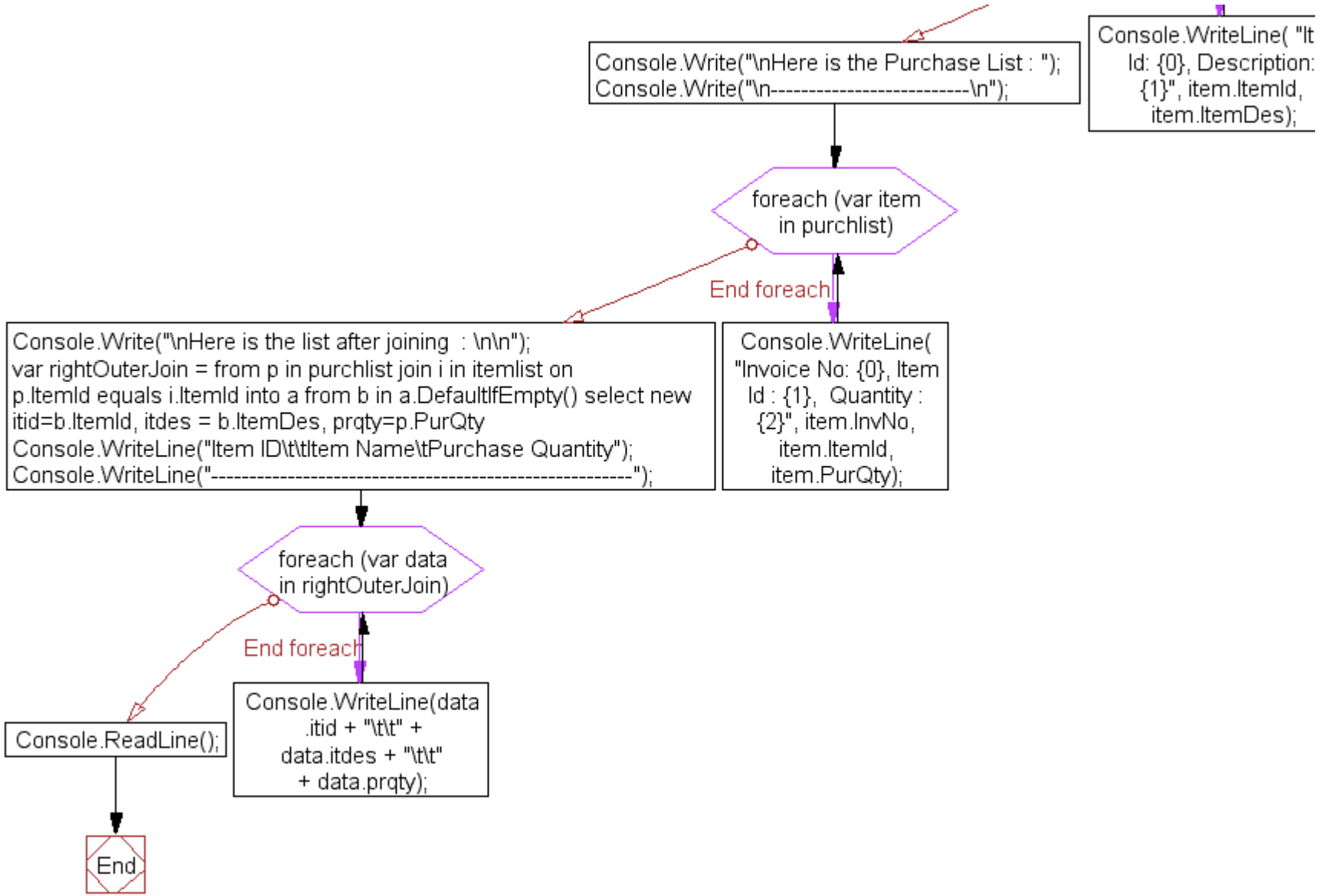
EKSEMPEL PROGRAM



EKSEMPEL PROGRAM



EKSEMPEL PROGRAM



LEKTIE

Kig på dette til næste gang

LEKTIE

- Læs:
- https://www.tutorialspoint.com/csharp/csharp_preprocessor_directives.htm
- https://www.tutorialspoint.com/csharp/csharp_regular_expressions.htm

Opgave

- Arbejd på jeres eksamensopgave

KILDER

Materiale benyttet i denne lektion
Noget af det er udover pensum-listen!

KILDER

Preprocessor Directives

- https://www.tutorialspoint.com/csharp/csharp_preprocessor_directives.htm

Regular Expressions

- https://www.tutorialspoint.com/csharp/csharp_regular_expressions.htm
- https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx

Debugging i Visual Studio

- https://msdn.microsoft.com/en-us/library/sc65sadd.aspx
- https://msdn.microsoft.com/en-us/library/y740d9d3.aspx

Eksempel

- http://www.w3resource.com/csharp-exercises/linq/csharp-linq-exercise-27.php