



**Code
Academy**



Dėstytojas

Edvinas Kesminas

Debug ir StringBuilder

Data



Šiandien išmoksite

01

Debug VisualStudio aplinkoje

02

Debug VisualStudio Sąsaja

03

Debug Break Points

04

Debug Immediate Window

05

StringBuilder

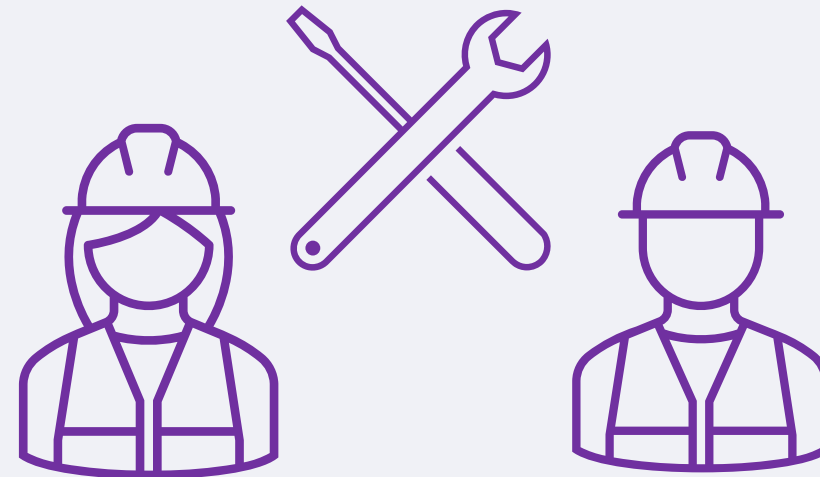
06

StopWatch Benchmark matavimas



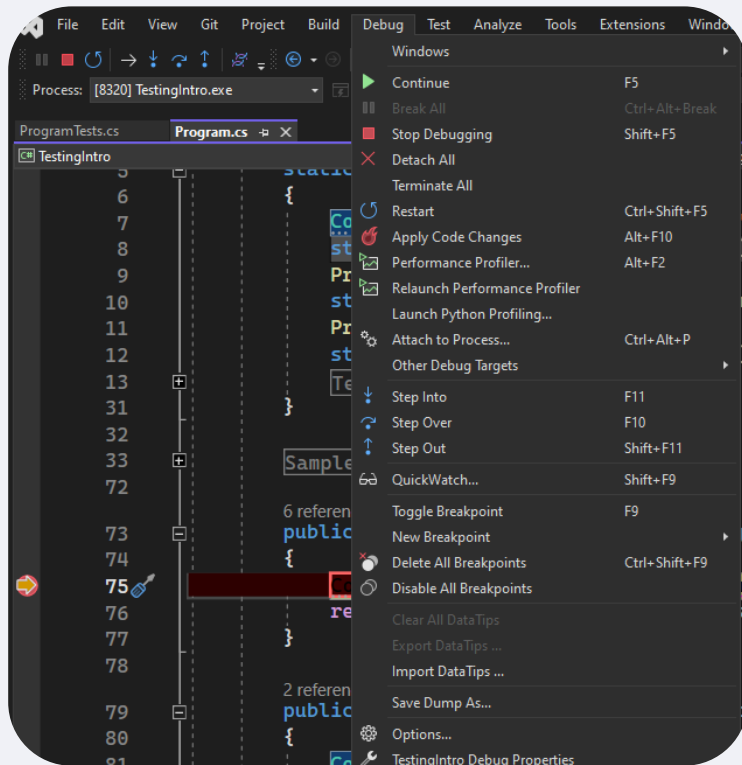
Debug VisualStudio aplinkoje

- Derinant ar programuojant programinę įrangą, yra neįmanoma išvengti klaidų ar netikėtų situacijų, kurios gali iškilti.
- Tokie įrankiai kaip "Visual Studio", tampa itin naudingi. "Visual Studio" siūlo galingą įrankių rinkinį klaidų šalinimui ir derinimui, įskaitant atgalinio derinimo (angl. "debugging") funkcijas.





Debug VisualStudio Sąsaja



- Atgalinio derinimo funkcijos "Visual Studio" leidžia programuotojui stebėti ir analizuoti vykdomą kodą žingsnis po žingsnio.
- Pagrindinis tikslas yra surasti ir ištaisyti klaidas, gerinant programos veikimą ir funkcionalumą.
- „Debugging“ su Visual Studio, mes galime „inspectinti“ kintamųjų reikšmes, laikinai stabdyti programą norimoje vietoje arba net išgauti tuometinių duomenų statistikas



Debug Break Points

- Derinant programinę įrangą, galima susidurti su situacijomis, kai reikia atidžiau stebėti vykdomą kodą.
- Šiuo atveju "break point" (taškas sustojimui) tampa naudingu įrankiu. "Break point" yra specifinė vieta kode, į kurią programa sustoja vykdymo metu, leidžiant programuotojui peržiūrėti kintamųjų reikšmes ir atlikti kitas veiksmų analizės funkcijas.
- Kai programa pasiekia "break point", jos vykdymas laikinai sustoja, o programuotojas gali tirti ir suprasti, kaip vyksta programos būsena ir kodas tuo metu.

```
73 public static string GetFullName(string firstName, string lastName)
74 {
75     Console.WriteLine("Registruotas naudotojas" + firstName + " " + lastName);
76     return firstName.Trim() + " " + lastName.Trim();
77 }
78
79 public static void PrintDanger(string text)
80 {
81     Console.BackgroundColor = ConsoleColor.Red;
82     Console.WriteLine(text);
83     Console.BackgroundColor = ConsoleColor.Red;
84 }
```

The screenshot shows a code editor with a dark theme. A break point (red dot) is set on line 75 of the `GetFullName` method. The code is in C#. The `Console.WriteLine` call on line 75 is highlighted in yellow. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 81 is highlighted in blue. The `Console.WriteLine(text);` line on line 82 is highlighted in green. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 83 is highlighted in red. The `Console.WriteLine(text);` line on line 84 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 85 is highlighted in red. The `Console.WriteLine(text);` line on line 86 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 87 is highlighted in red. The `Console.WriteLine(text);` line on line 88 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 89 is highlighted in red. The `Console.WriteLine(text);` line on line 90 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 91 is highlighted in red. The `Console.WriteLine(text);` line on line 92 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 93 is highlighted in red. The `Console.WriteLine(text);` line on line 94 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 95 is highlighted in red. The `Console.WriteLine(text);` line on line 96 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 97 is highlighted in red. The `Console.WriteLine(text);` line on line 98 is highlighted in blue. The `Console.BackgroundColor = ConsoleColor.Red;` line on line 99 is highlighted in red. The `Console.WriteLine(text);` line on line 100 is highlighted in blue.



Debug Immediate Window

```
Immediate Window
firstName
"Jonas"
firstName = "John"
"John"
firstName
"John"
```

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

- Visual Studio "Immediate Window" (tuo pačiu metu ir tiesioginio vykdymo langas) yra galingas ir patogus įrankis programuotojams derinant savo kodą.
- Tai yra interaktyvus langas, kuris leidžia tiesiogiai vykdyti C# kodo fragmentus ir peržiūrėti jų rezultatus.

```
6 references | 3/3 passing
public static string GetFullName(string firstName, string lastName)
{
    Console.WriteLine("Registruotas naudotojas" + firstName + " " + lastName);
    return firstName.Trim() + " " + lastName.Trim(); ≤ 1ms elapsed
}
```

C:\Users\Edvinas\source\repo X + v

```
2 references
publ Hello, Testing!
publ Registruotas naudotojasJohn Jonaitis
{
    Registruotas naudotojasJohn Jonaitis
    publ Hello, Testing!
```



Debug Logging

- Paprastas „loggingas“ yra svarbi praktika programuojant, kadangi padeda sekti ir užfiksuoti programos vykdymo informaciją.
- Tai yra procesas, kuriuo įrašomos svarbios žinutės, pranešimai arba duomenys apie programos būseną į tam tikrą žurnalizavimo sistemą.
- Paprastas „logginimas“ padeda programuotojui suprasti, kaip programa elgiasi ir pašalinti galimas klaidas.

The screenshot shows a Visual Studio code editor with a C# file named `TestCode`. The code defines a `Main` method that calls `GetFullName` and `PrintDanger` methods. The `PrintDanger` method is shown in a separate callout box. The debug console on the right shows the output of the program, including the text "Hello, Testing!" and several lines of log messages. The log messages are formatted with red text for "PIRMA IVESTIS" and black text for the rest of the message.

```
0 references
static void Main(string[] args)
{
    Console.WriteLine("Hello, Testing!");
    string JonasJonaitis = GetFullName("Jonas", "Jonaitis");
    PrintDanger("PIRMA IVESTIS");
    string PetrasPetraitis = GetFullName("Petras", "Jonaitis");
    PrintDanger("PIRMA IVESTIS");
    string GretaGretaite = GetFullName("Jonas", "Jonaitis");
}

public static void PrintDanger(string text)
{
    Console.BackgroundColor = ConsoleColor.Red;
    Console.WriteLine(text);
    Console.ResetColor();
}
```

Microsoft Visual Studio Debug Console

```
Hello, Testing!
Registruotas naudotojasJonas Jonaitis
PIRMA IVESTIS
Registruotas naudotojasPetras Jonaitis
PIRMA IVESTIS
Registruotas naudotojasJonas Jonaitis

C:\Users\Edvinas\source\repos\C# Repos\CA.NET\testingIntro.exe (process 34232) exited with code 0
To automatically close the console when debugging is complete, press Ctrl+Shift+F5 or the menu item Debug > Close All Console Windows.
```



Debug Unit Test Debugging

```
14 [TestMethod()]
15 | 0 references
16 public void GetFullName_AddCorrectNameSurname_ReturnsFullName()
17 {
18     // Arrange
19     string fakeName = "Jonas";
20     string fakeSurname = "Jonaitis";
21     string expected = "Jonas Jonaitis";
22
23     // Act
24     string actual = Program.GetFullName(fakeName, fakeSurname);
25
26     // Assert
27     Assert.AreEqual(expected, actual);
28 }
29
30 // Assert
31
32
```

- Kaip ir paprastą kodą, mes taip pat galime debugginti Unit Test'us jei atrandame, kad juose yra kažkokia problema.
- Juose galime naudoti break pointus ir analizės įrankius, kad suprasti kaip veikia mūsų testas.



Užduotis nr. 1

- Naudodami debuginimą raskite klaidą kode:

```
int a = 10;  
int b = 5;  
int c = 8;  
  
int max = a;  
if (b > max)  
    max = b;  
if (c > max)  
    max = b;  
  
Console.WriteLine("The maximum value is: " + max);
```

```
Console.WriteLine("The maximum value is: " + max);
```

```
max = 5;
```



Užduotis nr. 2

- Naudodami debuginimą raskite klaidą kode:

```
string firstName = "John";  
string lastName = "Doe";  
string fullName = firstName + lastName;  
Console.WriteLine("Full Name: " + fullName);  
...
```

```
Console.WriteLine("Full Name: " + firstName);
```



Užduotis nr. 3

- Programa turėtų skaičiuoti nuo 1 iki 10. Pataisykite naudodami debug:

```
int counter = 0;  
while (counter <= 10)  
{  
    Console.WriteLine("Count: " + counter);  
    counter--;  
}
```



Užduotis nr. 4

- Programa turėtų skaičiuoti nuo 1 iki 5. Pataisykite naudodami debug:

```
int i = 5;  
while (i > 0)  
{  
    Console.WriteLine(i);  
    i++;  
}
```



Užduotis nr. 5

- Programa bando palyginti du vardus. Pataisykite naudodami debug:

```
string name1 = "John";  
string name2 = "john";  
  
if (name1.Equals(name2))  
{  
    Console.WriteLine("Names are the same.");  
}  
else  
{  
    Console.WriteLine("Names are different.");  
}
```

```
Console.WriteLine("Names are different.");
```



StringBuilder

- Naudoti "StringBuilder" vietoj "string" yra naudinga tais atvejais, kai turite keisti arba sujungti daugybę tekstinių elementų.
- "StringBuilder" klasė leidžia efektyviai manipuluoti teksto eilutėmis, nes ji suteikia geresnį veikimo greitį ir mažesnę atminties naudojimo apimtį nei tiesioginės "string" manipuliacijos.
- Tai ypač pasireiškia, kai turite atlikti daug operacijų, tokias kaip teksto sujungimas arba žymų įterpimas, kad sukurtumėte galutinę tekstą.

```
StringBuilder stringBuilder = new StringBuilder();

stringBuilder.Append("Labas, ");
stringBuilder.Append("kaip sekasi? ");
stringBuilder.Append("Tikiuosi, kad viskas gerai!");

string result = stringBuilder.ToString();

Console.WriteLine(result);
```



StopWatch

```
Tiesioginės 'string' manipuliacijos: 00:00:01.4503927
Naudojant 'StringBuilder': 00:00:00.0004986
```

```
string text = "";
int iterations = 100000;

// Pirmasis scenarijus: Tiesioginės "string" manipuliacijos
Stopwatch stopwatch = new Stopwatch();
stopwatch.Start();

for (int i = 0; i < iterations; i++)
{
    text += "A";
}

stopwatch.Stop();
TimeSpan elapsedStringManipulation = stopwatch.Elapsed;

// Antrasis scenarijus: "StringBuilder" naudojimas
stopwatch.Reset();
stopwatch.Start();

StringBuilder stringBuilder = new StringBuilder();
for (int i = 0; i < iterations; i++)
{
    stringBuilder.Append("A");
}
string result = stringBuilder.ToString();

stopwatch.Stop();
TimeSpan elapsedStringBuilder = stopwatch.Elapsed;

// Išvesti rezultatus
Console.WriteLine("Tiesioginės 'string' manipuliacijos: " + elapsedStringManipulation);
Console.WriteLine("Naudojant 'StringBuilder': " + elapsedStringBuilder);
```

- "Stopwatch" klasė yra labai naudingas įrankis programuotojams matuoti laiką ir apskaičiuoti veikimo trukmę savo programose.
- Ji leidžia tiksliai ir patogiai matuoti laiką nuo tam tikro taško pradžios iki pabaigos. "Stopwatch" klasė suteikia galimybę matuoti labai mažus laiko tarpus



Užduotis nr. 6

- Parašykite programą, kuri gautą vartotojo įvestą eilutę ir naudodama StringBuilder, apverstų šią eilutę. Rodykite apverstą eilutę kaip rezultatą.
- Parašykite programą, kuri gautą vartotojo įvestą eilutę ir naudodama StringBuilder, pašalintų visus dublikatus iš šios eilutės. Rodykite pakeistą eilutę kaip rezultatą, be dublikatų simbolių.

```
Iveskite eilute: Labas pasauli sian labai grazus oras  
Apversta eilute: saro suzarg iabal nais iluasap sabal
```

```
Iveskite eilute: Labas pasauli  
Eilute be dublikatu: Labs puli
```




Užduotis nr. 7

- Tęsiame su 3-7 pamokos metodų aprašymus ir Unit test rašymus. Pabaigus pradedame rašyti 9-10 pamokų (out, ref ir for) užduočių Unit Testus.



Projektas nr. 1

- Savo 3 pamokos projekto funkcionalumą išjudinkite į atskirus metodus. Kurdami metodus bandykite stipriai akcentuoti testuojamą kodą (Turėtų kažkas grįžti iš pačio metodo). Parašę metodus juos visus padenkite testais. Kiekvienas scenarijus turėtų turėti nors 1 testą.



<https://learn.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs-2022>

<https://www.tutorialsteacher.com/articles/how-to-calculate-code-execution-time-in-csharp>

**Naudinga
informacija**