**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**UNIVERSITATEA DE STAT „ALECU RUSSO” DIN BĂLŢI**

**FACULTATEA DE ŞTIINŢE REALE, ECONOMICE ȘI ALE MEDIULUI**

**CATEDRA DE MATEMATICĂ ȘI INFORMATICĂ**

**UTILIZAREA INSTRUMENTELOR DE VERIFICARE ȘI APLICARE A STILULUI DE COD   
TEZĂ DE AN**

**Autor:**

Student al grupei IS21Z

**CAZACU-CONDRAT Dumitru**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**Conducător științific:**

**Skutnițki Olesea**

magistru, asist. univ.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**BĂLŢI, 2022**

**CUPRINS**

[INTRODUCERE 3](#_Toc103527872)

[1. CONCEPTELE DE BAZĂ A CODE STYLE-ULUI 4](#_Toc103527873)

[1.1. Noțiunea de code style. Conceptele de bază 4](#_Toc103527874)

[1.2. Importanța code style-ului în programare 4](#_Toc103527875)

[1.3. Principiile aplicării a unui Code Style 5](#_Toc103527876)

[1.4. Utilizarea Extensiilor pentru un Code Style Particular 8](#_Toc103527877)

[2. ANALIZA CODE STYLE-ULUI A PROGRAMLELOR ÎN LIMBAJUL DE PROGRAMARE C# ȘI APLICAREA INSTRUMENTELOR PENTRU STILIZAREA CODULUI 12](#_Toc103527878)

[2.1. Analiza și depistarea erorilor de stilizare a codului 12](#_Toc103527879)

[2.2. Aplicarea instrumentelor pentru înlăturarea erorilor 26](#_Toc103527880)

[2.3. Elaborarea unei extensii pentru stilizarea codului 33](#_Toc103527881)

[CONCLUZII 34](#_Toc103527882)

[BIBLIOGRAFIE 34](#_Toc103527883)

# INTRODUCERE

După o analiză a mai multor programe se poate de constatat că un programator bun trebuie neapărat să aibă un code style excelent. Astfel e ușor de a lucra în echipă și la distanță fără contact direct, în regim online. Chiar dacă cineva are talent de a programa fără un code style bun departe nu va putea ajunge.

Din această cauză, multe persoane care au început să programeze s-au întâlnit la început foarte des cu această problemă și le venea foarte greu să înțeleagă ce și cum trebuie să facă.

*Actualitatea studiului* – Analiza diferitor surse și a pieței IT denotă că pentru a-ți asigura un loc de muncă ca programator, în afara cunoașterii limbajului cerut este nevoie și de cunoașterea code style-ului.

*Scopul tezei de an* constă în cercetarea standardelor code style aplicate în limbajul de programare C#, utilizarea instrumentelor de verificare și aplicare a acestora pentru asigurarea lizibilității codului.

Pentru atingerea scopului au fost înaintate următoarele obiective:

* Studierea literaturii de specialitate la tema cercetării;
* Analiza codului în limbajul C# referitor la respectarea principiilor de code style;
* Utilizarea instrumentelor pentru înlăturarea erorilor de stilizare.

*Valoarea teoretică* – partea teoretică a lucrării date este destul de importantă, prin descrierea diferitor reguli și metode pentru crearea unui cod cât mai lizibil și plăcut de citit.

Teza de an cuprinde: introducere, 2 capitole, concluzii și o listă bibliografică**.** În primul capitol este prezentată informația generală despre code style în diferite limbaje, exemple bune și rele, la fel și o metodă de a simplifica programarea.

În al doilea capitol sunt cercetate 2 programe în limbajul C# unde va fi analizat codul după toate regulile. Teza este expusă pe 36 pagini, conține 7 imagini, o listă bibliografică din 15 surse bibliografice.

# CONCEPTELE DE BAZĂ A CODE STYLE-ULUI

## Noțiunea de code style. Conceptele de bază

Programming style, cunoscut și sub numele de code style, este un set de reguli sau linii directoare utilizate la scrierea codului sursă pentru un program la calculator. Se susține adesea că respectarea unui anumit stil de programare va ajuta programatorii să citească și să înțeleagă codul sursă conform stilului și să evite introducerea de erori. Stilurile de programare se ocupă în mod obișnuit de aspectul vizual al codului sursă, cu scopul de a fi lizibil. Software-ul a fost disponibil de mult timp care formatează codul sursă automat, lăsându-i pe programatori să se concentreze asupra denumirii, logicii și tehnicilor superioare. Ca punct practic, folosirea unui computer pentru a forma codul sursă economisește timp și este posibil să se impună apoi standardele la nivel de companie fără dezbateri [1].

Stilurile de programare sunt adesea concepute pentru un anumit limbaj de programare (sau familie de limbaje), stilul considerat bun în codul sursă C poate să nu fie considerat bun pentru codul sursă BASIC. Cu toate acestea, unele reguli sunt aplicate în mod obișnuit în multe limbaje.

# Importanța code style-ului în programare

Coding style este modul în care arată codul. Stilul de codare este extrem de personal și fiecare are propriul stil preferat. Fiecare își poate descoperi propriul stil personal analizând codul pe care l-a scris atunci când nu avea un ghid de stil la care să adere.

Fiecare are propriul stil datorită modului în care a învățat să scrie codul. Dacă cineva a folosit un mediu de dezvoltare integrat (IDE) precum Visual Studio pentru a învăța programare, atunci stilul se potrivește probabil cu cel impus de editor. Dacă însă pentru învățat s-a folosit un editor de text simplu, stilul probabil a evoluat din ceea ce începătorul credea că este mai lizibil. Nu numai editurile au nevoie de un ghid de stil. Dacă doriți să vă păstrați codul lizibil și ușor de întreținut chiar și la mulți ani după ce ați lansat un site web, un ghid de stil de codare este util și necesar.

S-ar putea chiar de observat că stilul se schimbă de la un limbaj la altul. Este posibil ca deciziile care se iau în JavaScript să nu fie transferate în CSS. De exemplu, cineva poate decide că șirurile JavaScript ar trebui să utilizeze ghilimele duble, în timp ce șirurile CSS ar trebui să utilizeze ghilimele simple. Acest lucru nu este neobișnuit, deoarece este prezentă tendința de a schimba contextul atunci când se comută înainte și înapoi între limbaje.

Cel mai important lucru atunci când se lucrează în echipă este comunicarea. Oamenii trebuie să poată lucra împreună eficient și singura modalitate de a face acest lucru este prin comunicare. În calitate de dezvoltatori, se comunică în primul rând prin cod. Se comunică cu alte părți ale software-ului prin cod și cu alți dezvoltatori tot prin cod. Dacă la cod se gândește ca la o modalitate de comunicare cu alți dezvoltatori, se poate de realizat că nu se scrie doar cod, ci se creează cod. Codul trebuie să-și comunice în mod clar și simplu scopul. Trebuie de reținut că codul este destinat să fie întreținut de altcineva decât o singură persoană într-o echipă și trebuie de comunicat cu toți membrii echipei.

Deseori începătorii în încercarea lor de a defini ce este code style-ul, și de a înțelege dacă este un detaliu important în programare, se întreabă dacă acest mic detaliu este într-adevăr atât de important. Răspunsul poate fi unul afirmativ și negativ în dependență de cum omul privește acest lucru.

* da, stilul codului nu contează cu adevărat pentru calculatorul care îl compilează;
* nu, micile detalii contează foarte mult pentru dezvoltatori care trebuie să întrețină codul.

Se poate de gândit la asta astfel: o singură greșeală de tipar într-o carte nu va perturba înțelegerea sau plăcerea lecturii. Cu toate acestea, dacă există multe greșeli de tipar, experiența de citire devine rapid enervantă pe măsură ce cititorul încearcă să descifreze sensul autorului, în ciuda cuvintelor folosite.

De aceea, este important ca echipa să dezvolte împreună un ghid de stil. Aducerea tuturor în aceeași cameră pentru a discuta despre ceea ce este cu adevărat important pentru echipă va avea ca rezultat o bază bună pentru ghidul de stil. Și ghidul de stil ar trebui să continue să crească pe măsură ce echipa se familiarizează mai mult între ei și cu software-ul la care lucrează [2].

## Principiile aplicării a unui Code Style

Există anumite reguli generale care se folosesc în toate limbajele pentru a avea un cod cât mai citeț. Iată câteva din ele:

* Fișierele trebuie păstrate având o cantitate mică, ceea ce se traduce la aproximativ 200 de linii per fișier (și nu mai mult de 500);
* Păstrarea fiecărui rând la mai puțin de 120 de caractere, de preferință mai puțin de 100;
* În POO variabilele de instanță ar trebui grupate într-un loc bine-cunoscut și ușor de găsit. În Java, variabilele se găsesc în partea de sus a paginii iar în C++, în partea de jos (O variabilă de instanță este o variabilă care este declarată într-o clasă, dar în afara constructorilor, metodelor sau blocurilor. Variabilele de instanță sunt create atunci când un obiect este creat și sunt accesibile tuturor constructorilor, metodelor sau blocurilor din clasă);
* Indentarea indică ierarhia și fiecare nivel (de exemplu, nume de metode, blocuri) ar trebui să fie unul indentat la dreapta într-un fișier [3].

Un cod corespunde normelor code style-ului dacă sunt respectate următoarele cerințe:

* Respectarea Identării Construcțiilor;
* Aplicarea Alinierii verticale;
* Utilizarea spațiilor libere;
* Utilizarea Filelor.

Stilurile de indentare ajută la identificarea fluxului de control și a blocurilor de cod. În unele limbaje de programare, indentarea este folosită pentru a delimita blocurile logice de cod; indentarea corectă în aceste cazuri este mai mult decât o chestiune de stil. În alte limbi, indentarea și spațiul alb nu afectează funcția, deși indentarea logică și consecventă face codul mai ușor de citit. Exemplu:

***if*** *(hours < 24 && minutes < 60 && seconds < 60) {*

***return*** *true;*

*}* ***else*** *{*

***return*** *false;*

*}*

sau

***if*** *(hours < 24 && minutes < 60 && seconds < 60)*

*{*

***return*** *true;*

*}*

***else***

*{*

***return*** *false;*

*}*

În comparație cu:

***if*** *( hours < 24*

*&& minutes < 60*

*&& seconds < 60*

*)*

*{****return*** *true*

*;}* ***else***

*{****return*** *false*

*;}*

Primele două exemple sunt probabil mult mai lizibile deoarece sunt indentate într-un mod stabilit. Acest stil de indentare este util în special atunci când sunt folosite mai multe bucle.

Este adesea util de aliniat elemente similare pe verticală, pentru a face mai evidente erorile generate de greșeli de scriere.

*$search =* ***array****('a', 'b', 'c', 'd', 'e');*

*$replacement =* ***array****('foo', 'bar', 'baz', 'quux');*

*// Another example:*

*$value = 0;*

*$anothervalue = 1;*

*$yetanothervalue = 2;*

*$search =* ***array****('a', 'b', 'c', 'd', 'e');*

*$replacement =* ***array****('foo', 'bar', 'baz', 'quux');*

*// Another example:*

*$value = 0;*

*$anothervalue = 1;*

*$yetanothervalue = 2;*

Ultimul exemplu clarifică intuitiv un lucru care nu era clar în primul:

* există un termen de căutare mai mult decât există termeni de înlocuire.

Dacă aceasta este o eroare, acum este mai probabil să fie depistată.

Cu toate acestea, e important de reținut că există și minusuri:

* Fragilitate: dacă un programator nu formatează corect tabelul atunci când face o modificare, atunci rezultatul devine o mizerie care se deteriorează odată cu alte astfel de modificări. Operațiile simple cum ar fi căutarea și înlocuirea, pot, de asemenea, să distrugă formatarea.
* Rezistența la modificare: formatarea tabelară necesită mai mult efort de întreținut. Acest lucru poate descuraja un programator să facă o schimbare benefică, cum ar fi adăugarea, corectarea sau îmbunătățirea numelui unui identificator, deoarece va strica formatarea.
* Încrederea pe fontul mono spațiat: formatarea tabelară presupune că editorul folosește un font cu lățime fixă. Mulți editori de cod modern acceptă fonturi proporționale, iar programatorul poate prefera să folosească un font proporțional pentru lizibilitate.
* Dependență de unealtă: o parte din efortul de menținere a alinierii poate fi atenuat de instrumente, deși acest lucru creează o dependență de astfel de instrumente. [1]

În acele situații în care este necesar un spațiu alb, gramaticile majorității limbilor cu format liber nu sunt preocupate de cantitatea care apare. Stilul legat de spațiul alb este folosit în mod obișnuit pentru a îmbunătăți lizibilitatea. În prezent, nu sunt cunoscute fapte concrete (concluzii din studii) despre care dintre stilurile de spații albe au cea mai bună lizibilitate. Fiecare are preferințele sale, care pot fi arătate în următoarele exemple de cod scris în limbajul C:

*int i;*

***for****(i=0;i<10;++i){*

*printf("%d",i\*i+i);*

*}*

*int i;*

***for*** *(i = 0; i < 10; ++i) {*

*printf("%d", i \* i + i);*

*}*

Utilizarea filelor pentru a crea spațiu alb prezintă probleme speciale atunci când nu se acordă suficientă atenție deoarece locația punctului de tabulare poate fi diferită în funcție de instrumentele utilizate și chiar de preferințele utilizatorului. De exemplu, un programator preferă tabulaturi de patru și are setul de instrumente configurat în acest fel și le folosește pentru a-și formata codul. [1]

*int ix; // Index to scan array*

*long sum; // Accumulator for sum*

*int ix; // Index to scan array*

*long sum; // Accumulator for sum*

O soluție utilizată pe scară largă la această problemă implică interzicerea utilizării fil-urilor pentru aliniere sau reguli privind modul în care trebuie setate punctele de filare. Trebuie de reținut că filele funcționează bine cu condiția să fie utilizate în mod consecvent, limitate la indentarea logică și să nu fie utilizate pentru aliniere. [1]

Exemplu:

***class******MyClass*** *{*

*int foobar(*

*int qux, // first parameter*

*int quux); // second parameter*

*int foobar2(*

*int qux, // first parameter*

*int quux, // second parameter*

*int quuux); // third parameter*

*};*

## Utilizarea Extensiilor pentru un Code Style Particular

Pentru simplifica programarea există extensii care ajută programatorii să aibă un code style unit în diferite limbaje de programare. Unul dintre cel mai popular folosit pe larg în Visual Studio Code (care e un editor de cod realizat de Microsoft pentru Windows, Linux și macOS) este: Prettier – Code formater realizată în anul 2016. În marea majoritate IDE-urile (care este un mediu de dezvoltare integrat) precum Visual Studio au deja încorporat în programă un formator foarte bun și nu este necesar de a instala unul.

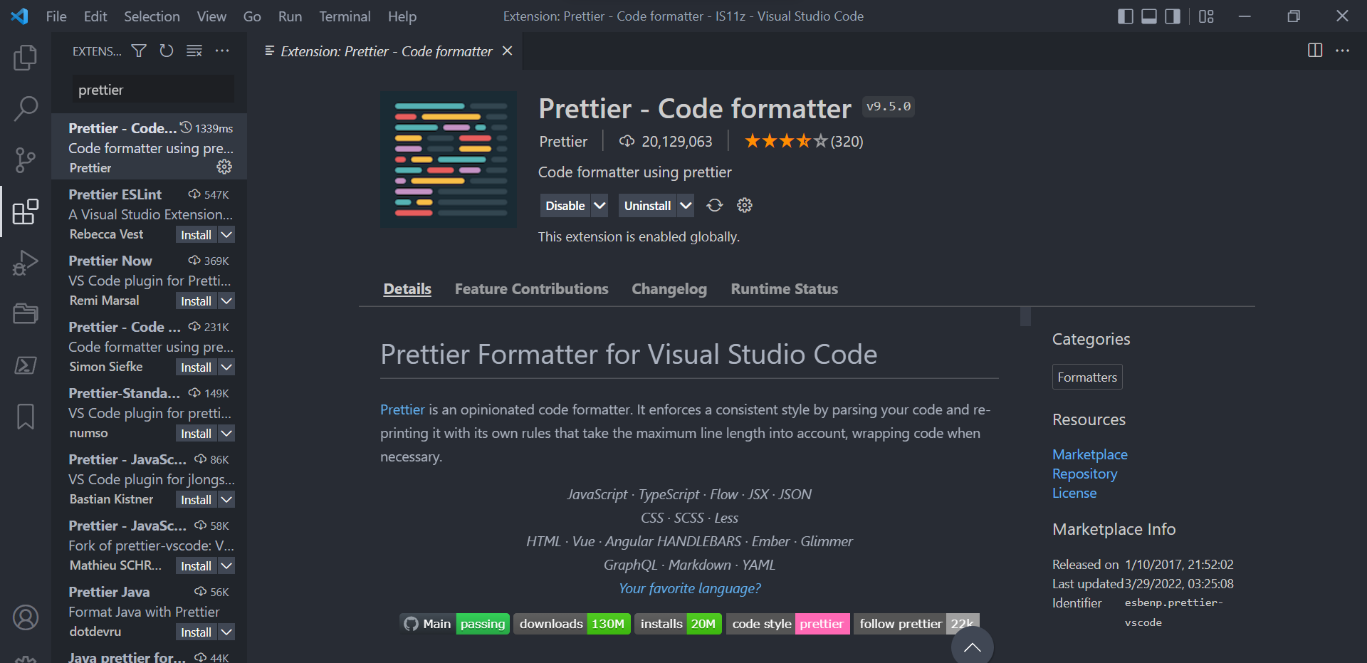
Pentru a instala extensia dată este necesar de a apăsa pe butonul cu extensii și de căutat extensia Prettier, după care se alege primul din listă pentru instalare cum este arătat în figura 1.1 [4].

Figura 1.1. Instalarea extensiei Prettier – Code formater.

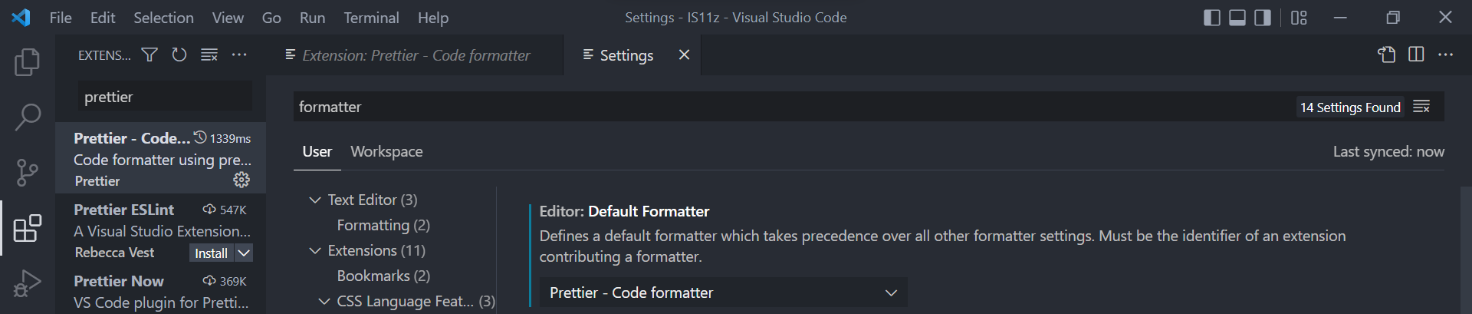
După aceasta este nevoie de a intra în setări pentru pune ca formatarea dată să fie cea implicită folosită în toate limbajele cum este arătat în figura 1.2.

Figura 1.2. Intrarea în setări pentru a pune extensia ca implicită.

La fel pentru comoditate se poate de pornit și opțiunea de formatare când se salvează fișierul pentru a economisi timp și de a ușura experiența începătorilor cum este vizibil în figura 1.3.

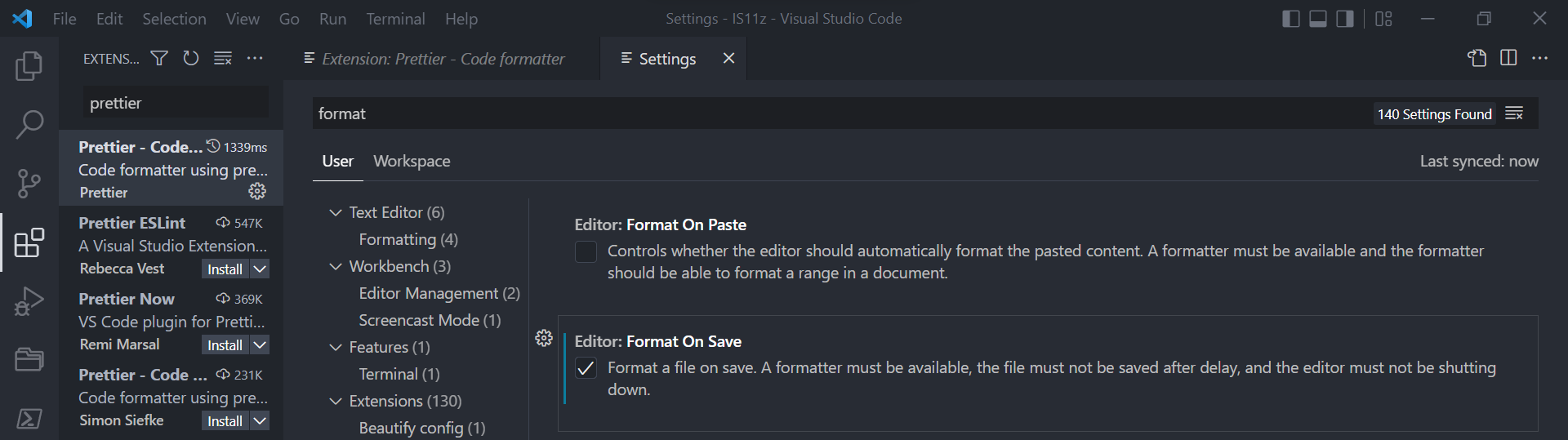


Figura 1.3. Activarea Format on save.

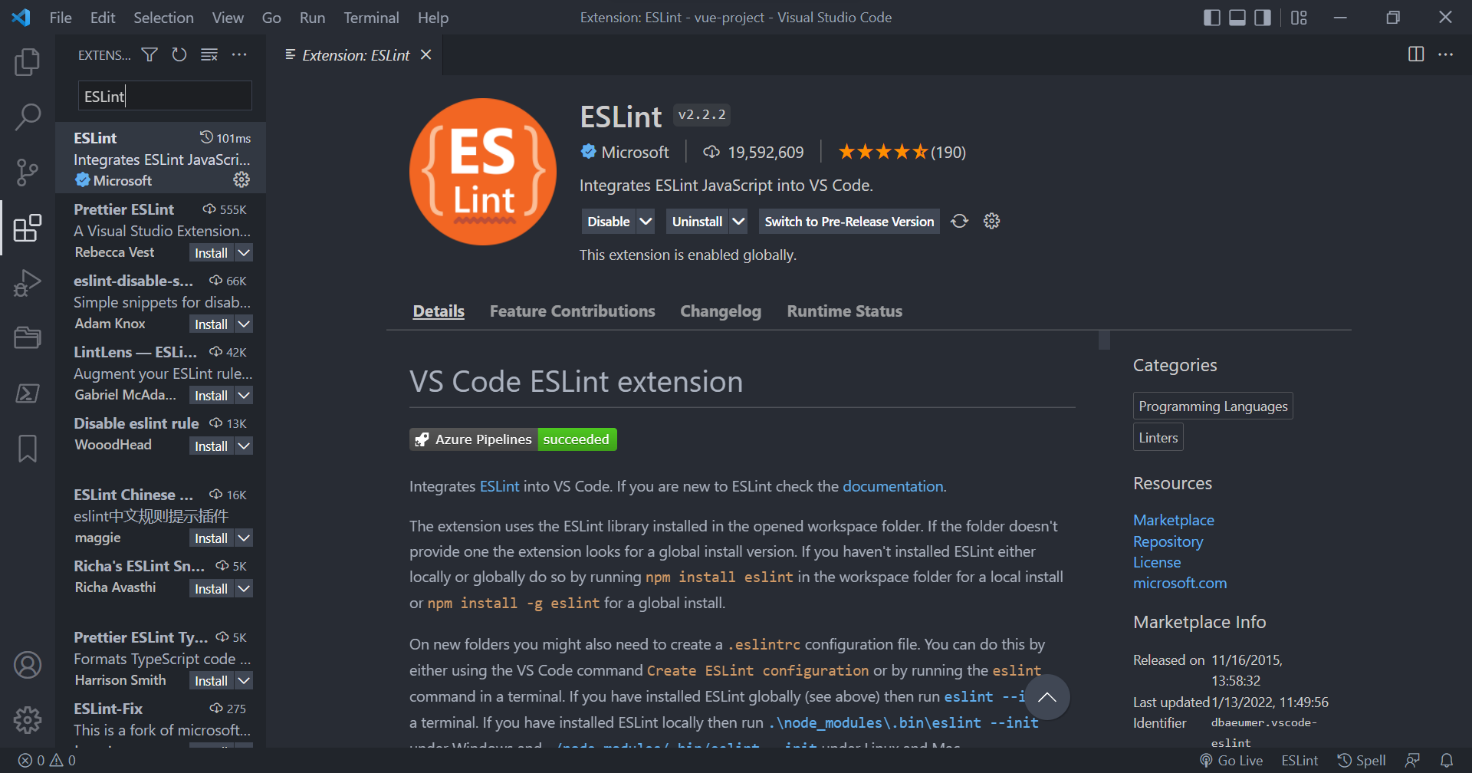
O altă extensie populară este ESLint care a fostă creată în anul 2013 înaintea extensiei Prettier. ESLint analizează codul complet ceea ce însemnă că nu numai formatează codul ci și arată greșelile logice care duc la erori. Pentru a instala această extensie la fel în meniul cu extensii se caută deja ESLint, apoi se descarcă cum este arătat în figura 1.4 [5].

Figura 1.4. Instalarea extensiei ESLint.

Însă apare întrebarea care din aceste 2 este cea mai bună. Vor fi analizate câteva diferențe:

* ESlint nu este doar un formator de cod, ci ajută dezvoltatorii să găsească erori de codare. ESLint va da un avertisment dacă se utilizează o variabilă fără a o declara. Prettier nu are o asemenea abilitate;
* De asemenea, ESLint va anunța ce este în neregulă cu formatarea codului și va oferi opțiuni pentru a remedia problema [6].

Majoritatea dezvoltatorilor folosesc ambele extensii. Însă acesta are un mare neajuns și anume că la proiecte mare viteza de lucru va fi mult mai scăzută dacă calculatorul nu este destul de performant.

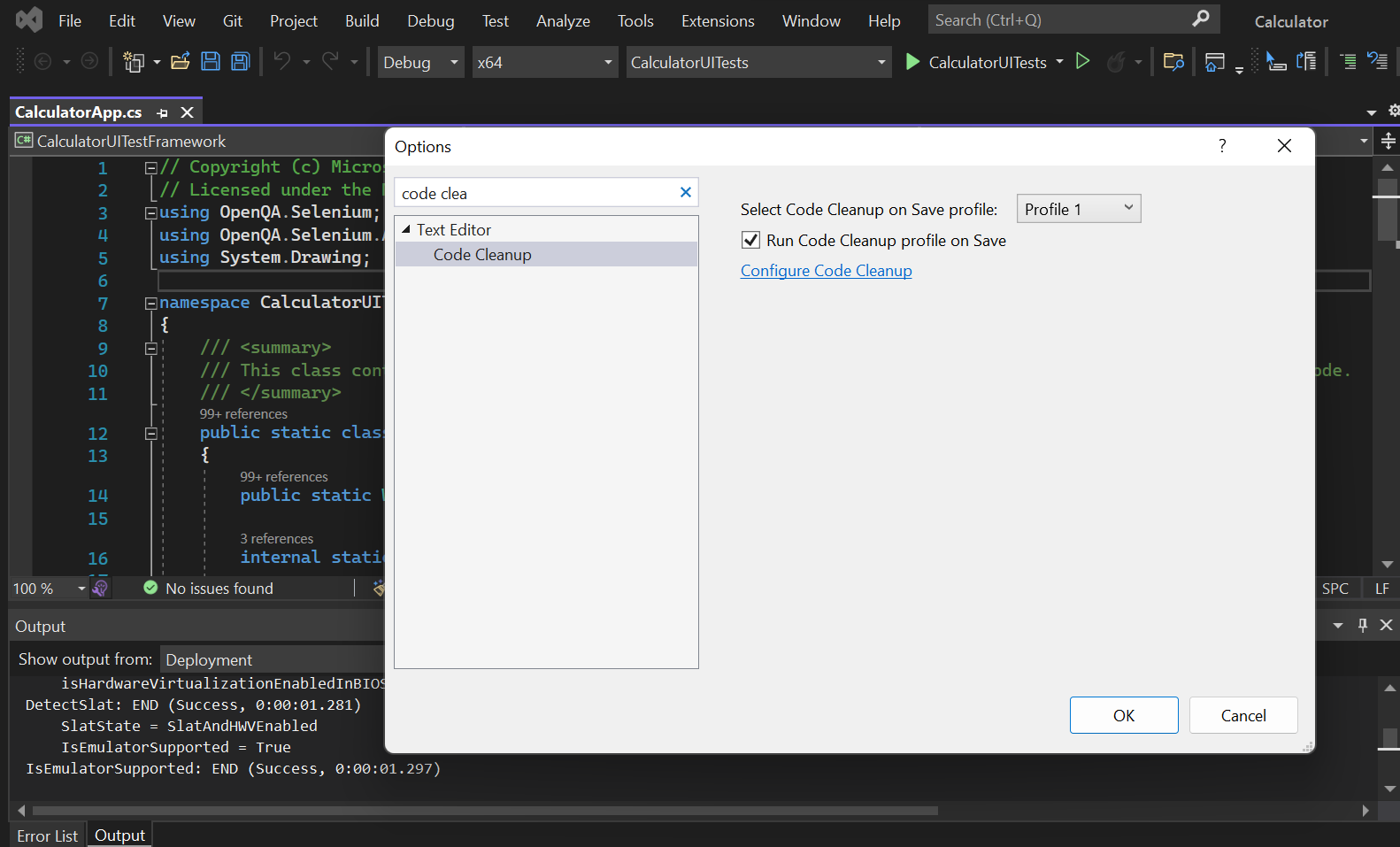
La fel este posibil de a activa funcția format on save în IDE-ul Visual Studio. Pentru aceasta în tools trebuie de intrat în opțiuni și aici se găsește code cleanup cum este arătat în figura 1.5.

Figura 1.5. Activarea format on save în IDE-ul Visual Studio.

În același este posibil de a face și niște configurări pentru diferite opțiuni spre exemplu culorile parantezelor. Pentru aceasta tot aici se găsește meniul de configurare unde se va deschide o altă fereastră. Trebuie de ținut cont că nu se recomandă pentru un începător de a configura cum va arăta codul întru-un IDE, este recomandat ca să folosească opțiunile implicite. În figura 1.6. este arătat cum se poate de intrat în meniul de configurare.

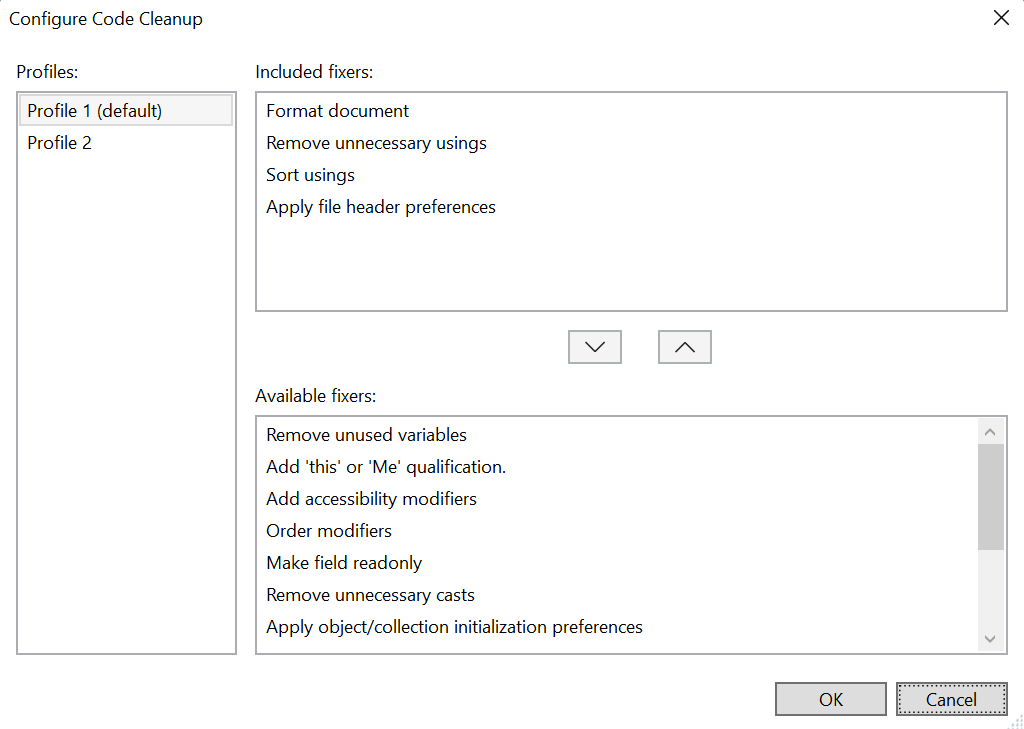


Figura 1.6. Intrarea în meniul de configurare în IDE-ul Visual Studio

# ANALIZA CODE STYLE-ULUI A PROGRAMLELOR ÎN LIMBAJUL DE PROGRAMARE C# ȘI APLICAREA INSTRUMENTELOR PENTRU STILIZAREA CODULUI

## Analiza și depistarea erorilor de stilizare a codului

Prima aplicație care va fi analizată este calculatorul din Windows 10/11 creată și întreținută de compania Microsoft. Acesta este o aplicație esențială și majoritatea oamenilor se folosesc de ea zi de zi. La început se va lua și analiza codul în IDE-ul Visual Studio apoi va fi comparat cu codul scris de începător.

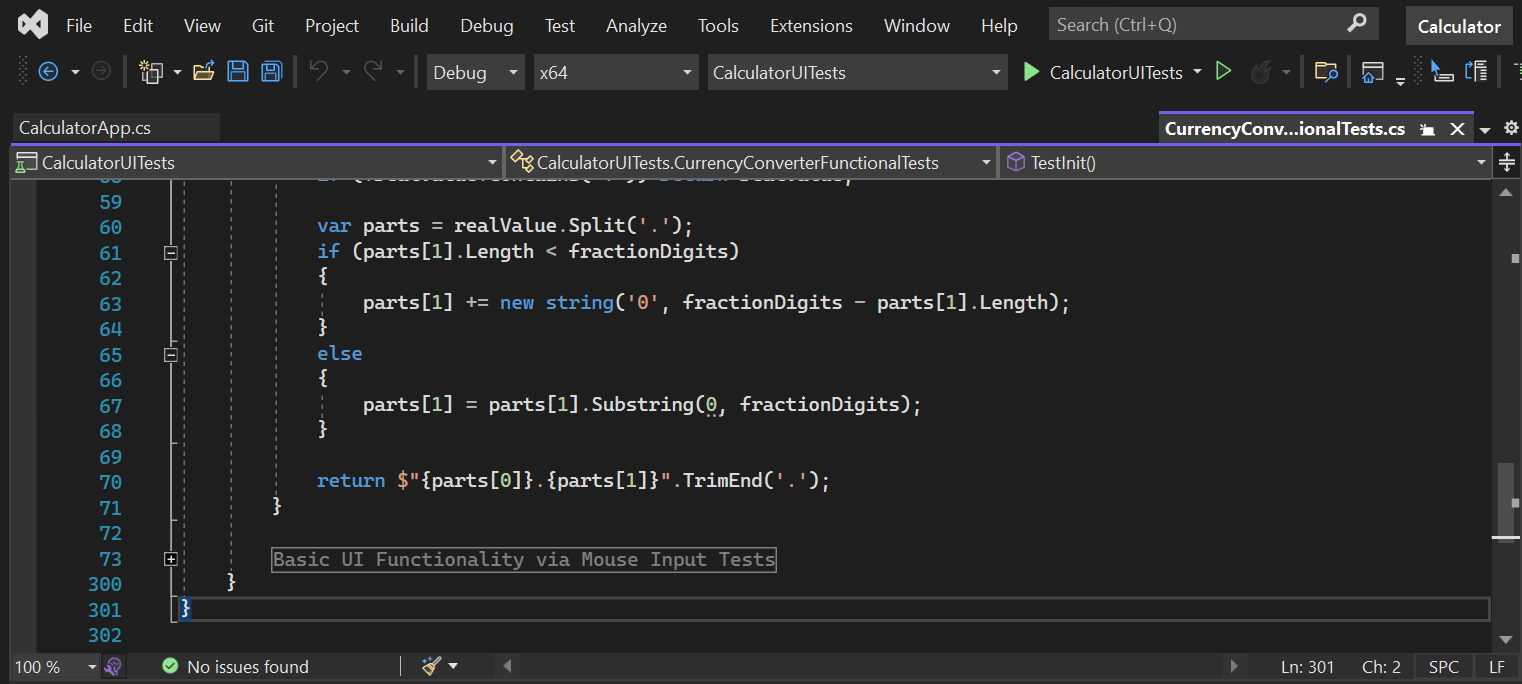
Primul lucru care se observă deodată este faptul că se respectă 2 din regulile generale și anume: fișierele trebuie păstrate având o cantitate mică, ceea ce se traduce la aproximativ 200 de linii per fișier (și nu mai mult de 500) și păstrarea fiecărui rând la mai puțin de 120 de caractere, de preferință mai puțin de 100 cum este vizibil în figura 2.1.

Figura 2.1. Respectarea regulii generale de număr a liniilor pe fișier.

*// Copyright (c) Microsoft Corporation. All rights reserved.*

*// Licensed under the MIT License.*

*using OpenQA.Selenium;*

*using OpenQA.Selenium.Appium.Windows;*

*using System.Drawing;*

*namespace CalculatorUITestFramework*

*{*

*/// <summary>*

*/// This class contains the UI automation objects and helper methods available when the Calculator is in any mode.*

*/// </summary>*

*public static class CalculatorApp*

*{*

*public static WindowsElement Window => session.FindElementByClassName("Windows.UI.Core.CoreWindow");*

*internal static WindowsElement Header => session.TryFindElementByAccessibilityId("Header");*

*private static WindowsDriver<WindowsElement> session => WinAppDriver.Instance.CalculatorSession;*

*private static WindowsElement AppName => session.TryFindElementByAccessibilityId("AppName");*

*/// <summary>*

*/// Gets the text from the Header*

*/// </summary>*

*/// <returns>The string shown in the UI.</returns>*

*public static string GetCalculatorHeaderText()*

*{*

*return Header.Text;*

*}*

*///// <summary>*

*///// Clicks the AppName element on Windows Calculator to ensure that the app has focus*

*///// </summary>*

*public static void EnsureCalculatorHasFocus()*

*{*

*AppName.Click();*

*}*

*/// <summary>*

*/// Click the window (to lose focus on components)*

*/// </summary>*

*public static void ClickOnWindow()*

*{*

*Window.Click();*

*}*

*/// <summary>*

*/// If the the Dock Panel for the History and Memory lists is not displayed, resize the window*

*/// Two attempts are made, the the lable is not found a "not found" exception is thrown*

*/// </summary>*

*public static void ResizeWindowToDisplayMemoryHistoryDockPanel()*

*{*

*// Put the calculator in the upper left region of the screen*

*WinAppDriver.Instance.CalculatorSession.Manage().Window.Position = new Point(8, 8);*

*GrowWindowToShowDock(WinAppDriver.Instance.CalculatorSession.Manage().Window.Size.Width);*

*}*

*/// <summary>*

*/// Increases the size of the window until Dock Panel for the History/Memory panel is visible*

*/// </summary>*

*private static void GrowWindowToShowDock(int width)*

*{*

*if (width > 2100)*

*{*

*throw new NotFoundException("Could not the Dock Panel for the History and Memory lists");*

*}*

*if (!session.PageSource.Contains("DockPanel"))*

*{*

*var height = WinAppDriver.Instance.CalculatorSession.Manage().Window.Size.Height;*

*WinAppDriver.Instance.CalculatorSession.Manage().Window.Size = new Size(width, height);*

*//give window time to render new size*

*System.Threading.Thread.Sleep(10);*

*GrowWindowToShowDock(width + 100);*

*}*

*}*

*}*

*}[7]*

Se observă faptul că secvența dată de cod este impecabilă. Ulterior secvența va fi comparată cu secvența elaborată de un programator începător.

*using System;*

*using System.Collections.Generic;*

*using System.ComponentModel;*

*using System.Data;*

*using System.Drawing;*

*using System.Linq;*

*using System.Text;*

*using System.Threading.Tasks;*

*using System.Windows.Forms;*

*namespace Calculator*

*{*

*public partial class Form1 : Form*

*{Double Result\_Value = 0;*

*String Operator\_Performed = " ";*

*bool PerformedOp = false;*

*public Form1()*

*{InitializeComponent();*

*}*

*private void Form1\_Load(object sender, EventArgs e)*

*{*

*}*

*private void button15\_Click(object sender, EventArgs e)*

*{*

*// numbers button and point*

*if (textBox\_Result.Text == "0" || PerformedOp)*

*textBox\_Result.Clear();*

*PerformedOp = false;*

*Button button = (Button)sender;*

*if(button.Text == ".")*

*{*

*if(!textBox\_Result.Text.Contains("."))*

*textBox\_Result.Text += button.Text;*

*}*

*else*

*textBox\_Result.Text += button.Text;*

*}*

*private void label1\_Click(object sender, EventArgs e)*

*{*

*}*

*private void Operator\_click\_Event(object sender, EventArgs e)*

*{*

*// +, -, \*, / operators*

*Button button = (Button)sender;*

*if (Result\_Value != 0)*

*{ button16.PerformClick();*

*Operator\_Performed = button.Text;*

*label\_Show\_Op.Text = Result\_Value + " " + Operator\_Performed;*

*PerformedOp = true;*

*}*

*else*

*{ Operator\_Performed = button.Text;*

*Result\_Value = Double.Parse(textBox\_Result.Text);*

*label\_Show\_Op.Text = Result\_Value + " " + Operator\_Performed;*

*PerformedOp = true;*

*}*

*}*

*private void button5\_Click(object sender, EventArgs e)*

*{*

*//CLEAR ENTRY BUTTON*

*textBox\_Result.Text = "0";*

*}*

*private void button6\_Click(object sender, EventArgs e)*

*{*

*//CLEAR BUTTON*

*textBox\_Result.Text = "0";*

*Result\_Value = 0;*

*label\_Show\_Op.Text = " ";*

*}*

*private void button16\_Click(object sender, EventArgs e)*

*{*

*// EQUALS BUTTON*

*switch (Operator\_Performed)*

*{*

*case "+":*

*textBox\_Result.Text = (Result\_Value + Double.Parse(textBox\_Result.Text)).ToString();*

*break;*

*case "-":*

*textBox\_Result.Text = (Result\_Value - Double.Parse(textBox\_Result.Text)).ToString();*

*break;*

*case "\*":*

*textBox\_Result.Text = (Result\_Value \* Double.Parse(textBox\_Result.Text)).ToString();*

*break;*

*case "/":*

*textBox\_Result.Text = (Result\_Value / Double.Parse(textBox\_Result.Text)).ToString();*

*break;*

*default:*

*break;*

*}*

*Result\_Value = Double.Parse(textBox\_Result.Text);*

*label\_Show\_Op.Text = " ";*

*} }} [8]*

Se observă deodată faptul că la fel se respectă 2 din regulile generale: fișierele trebuie păstrate având o cantitate mică, ceea ce se traduce la aproximativ 200 de linii per fișier (și nu mai mult de 500) și păstrarea fiecărui rând la mai puțin de 120 de caractere, de preferință mai puțin de 100. Însă sunt prezente multe alte greșeli de stil. Nu se folosesc peste tot filele și spațiile libere, iar în locurile folosite sunt folosite greșit. La fel indentarea nu este prezentă peste tot.

Mai departe va fi analizat codul companiei Microsoft care a făcut un notepad pentru a lucra cu fișierele XML și va fi comparat cu cel al unui începător care a scris un notepad simplu.

*using System;*

*using System.Runtime.InteropServices;*

*using System.Text;*

*using System.Drawing;*

*using System.Windows.Forms;*

*using System.Diagnostics;*

*using System.Threading;*

*namespace UnitTests*

*{*

*// Why the heck does .NET provide SendKeys but not mouse simulation???*

*// Another interesting tid-bit. Reading the cursor position doesn't work over*

*// terminal server!*

*public class Mouse*

*{*

*static int Timeout = 100;*

*private Mouse() { }*

*private static MouseFlags AddMouseDownFlags(MouseFlags flags, MouseButtons buttons)*

*{*

*if ((buttons & MouseButtons.Left) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_LEFTDOWN;*

*}*

*if ((buttons & MouseButtons.Right) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_RIGHTDOWN;*

*}*

*if ((buttons & MouseButtons.Middle) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_MIDDLEDOWN;*

*}*

*if ((buttons & MouseButtons.XButton1) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_XDOWN;*

*}*

*return flags;*

*}*

*private static MouseFlags AddMouseUpFlags(MouseFlags flags, MouseButtons buttons)*

*{*

*if ((buttons & MouseButtons.Left) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_LEFTUP;*

*}*

*if ((buttons & MouseButtons.Right) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_RIGHTUP;*

*}*

*if ((buttons & MouseButtons.Middle) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_MIDDLEUP;*

*}*

*if ((buttons & MouseButtons.XButton1) != 0)*

*{*

*flags |= MouseFlags.MOUSEEVENTF\_XUP;*

*}*

*return flags;*

*}*

*public static void MouseDown(Point pt, MouseButtons buttons)*

*{*

*MouseMoveTo(pt.X, pt.Y);*

*MouseInput input = GetAbsoluteMouseInput(pt.X, pt.Y);*

*MouseFlags flags = AddMouseDownFlags((MouseFlags)input.dwFlags, buttons);*

*input.dwFlags = (int)(flags | MouseFlags.MOUSEEVENTF\_ABSOLUTE);*

*SendInput(input);*

*}*

*public static void MouseUp(Point pt, MouseButtons buttons)*

*{*

*MouseInput input = GetAbsoluteMouseInput(pt.X, pt.Y);*

*MouseFlags flags = AddMouseUpFlags((MouseFlags)input.dwFlags, buttons);*

*input.dwFlags = (int)(flags | MouseFlags.MOUSEEVENTF\_ABSOLUTE);*

*SendInput(input);*

*}*

*public static void MouseClick(Point pt, MouseButtons buttons)*

*{*

*MouseDown(pt, buttons);*

*MouseUp(pt, buttons);*

*}*

*public static void MouseDoubleClick(Point pt, MouseButtons buttons)*

*{*

*MouseClick(pt, buttons);*

*Thread.Sleep(1);*

*MouseClick(pt, buttons);*

*}*

*private static MouseInput GetAbsoluteMouseInput(int x, int y)*

*{*

*MouseInput input = new MouseInput();*

*input.type = (int)(InputType.INPUT\_MOUSE);*

*input.dx = x;*

*input.dy = y;*

*input.dwFlags = (int)(MouseFlags.MOUSEEVENTF\_MOVE\_NOCOALESCE | MouseFlags.MOUSEEVENTF\_ABSOLUTE);*

*input.mouseData = 0;*

*input.time = 0;*

*return input;*

*}*

*private static MouseInput GetVirtualMouseInput(int x, int y)*

*{*

*MouseInput input = new MouseInput();*

*input.type = (int)(InputType.INPUT\_MOUSE);*

*input.dx = (x \* 65536) / GetSystemMetrics(SM\_CXSCREEN);*

*input.dy = (y \* 65536) / GetSystemMetrics(SM\_CYSCREEN);*

*input.dwFlags = (int)(MouseFlags.MOUSEEVENTF\_MOVE\_NOCOALESCE | MouseFlags.MOUSEEVENTF\_VIRTUALDESK);*

*input.mouseData = 0;*

*input.time = 0;*

*return input;*

*}*

*public static void MouseMoveTo(int x, int y)*

*{*

*MouseInput input = GetAbsoluteMouseInput(x, y);*

*MouseFlags flags = (MouseFlags)input.dwFlags;*

*input.dwFlags = (int)flags;*

*SendInput(input);*

*Application.DoEvents();*

*}*

*const int DragDelayDrop = 200;*

*public static void MouseDragDrop(Point start, Point end, int step, MouseButtons buttons)*

*{*

*int s = Timeout;*

*Timeout = 10;*

*MouseDown(start, buttons);*

*Application.DoEvents();*

*Thread.Sleep(DragDelayDrop);*

*MouseDragTo(start, end, step);*

*Thread.Sleep(DragDelayDrop);*

*MouseUp(end, buttons);*

*Application.DoEvents();*

*Thread.Sleep(DragDelayDrop);*

*Timeout = s;*

*}*

*public static void MouseMoveTo(Point start, Point end, int step)*

*{*

*MouseDragTo(start, end, step);*

*}*

*public static void MouseDragTo(Point start, Point end, int step)*

*{*

*// Interpolate and move mouse smoothly over to given location.*

*double dx = end.X - start.X;*

*double dy = end.Y - start.Y;*

*int length = (int)Math.Sqrt((dx \* dx) + (dy \* dy));*

*step = Math.Abs(step);*

*for (int i = 0; i < length; i += step)*

*{*

*int tx = start.X + (int)((dx \* i) / length);*

*int ty = start.Y + (int)((dy \* i) / length);*

*MouseMoveTo(tx, ty);*

*Thread.Sleep(1);*

*Application.DoEvents();*

*}*

*MouseMoveTo(end.X, end.Y);*

*}*

*public static void MouseWheel(AutomationWrapper w, int clicks)*

*{*

*var c = Cursor.Position;*

*if (w != null)*

*{*

*c = w.PhysicalToLogicalPoint(c);*

*}*

*MouseInput input = GetAbsoluteMouseInput(c.X, c.Y);*

*input.mouseData = clicks;*

*input.dwFlags |= (int)MouseFlags.MOUSEEVENTF\_WHEEL;*

*SendInput(input);*

*}*

*static void SendInput(MouseInput input)*

*{*

*//Trace.WriteLine("SendInput:" + input.dx + "," + input.dy + " cursor is at " + Cursor.Position.X + "," + Cursor.Position.Y);*

*if ((input.dwFlags & (int)MouseFlags.MOUSEEVENTF\_ABSOLUTE) != 0 &&*

*(input.dwFlags & (int)MouseFlags.MOUSEEVENTF\_VIRTUALDESK) == 0)*

*{*

*Cursor.Position = new Point(input.dx, input.dy);*

*}*

*Debug.WriteLine("SendInput x={0}, y={1}, flags={2:x}", input.dx, input.dy, input.dwFlags);*

*input.time = Environment.TickCount & Int32.MaxValue;*

*int cb = Marshal.SizeOf(input);*

*Debug.Assert(cb == 28); // must match what C++ returns for the INPUT union.*

*IntPtr ptr = Marshal.AllocCoTaskMem(cb);*

*try*

*{*

*Marshal.StructureToPtr(input, ptr, false);*

*uint rc = SendInput(1, ptr, cb);*

*if (rc != 1)*

*{*

*int hr = GetLastError();*

*throw new ApplicationException("SendInput error " + hr);*

*}*

*}*

*finally*

*{*

*Marshal.FreeCoTaskMem(ptr);*

*}*

*}*

*[DllImport("Kernel32.dll")]*

*static extern int GetLastError();*

*// Simluate MouseEvents*

*enum InputType { INPUT\_MOUSE = 0, INPUT\_KEYBOARD = 1, INPUT\_HARDWARE = 2 };*

*enum MouseFlags*

*{*

*MOUSEEVENTF\_MOVE = 0x0001, /\* mouse move \*/*

*MOUSEEVENTF\_LEFTDOWN = 0x0002, /\* left button down \*/*

*MOUSEEVENTF\_LEFTUP = 0x0004, /\* left button up \*/*

*MOUSEEVENTF\_RIGHTDOWN = 0x0008, /\* right button down \*/*

*MOUSEEVENTF\_RIGHTUP = 0x0010, /\* right button up \*/*

*MOUSEEVENTF\_MIDDLEDOWN = 0x0020, /\* middle button down \*/*

*MOUSEEVENTF\_MIDDLEUP = 0x0040, /\* middle button up \*/*

*MOUSEEVENTF\_XDOWN = 0x0080, /\* x button down \*/*

*MOUSEEVENTF\_XUP = 0x0100, /\* x button down \*/*

*MOUSEEVENTF\_WHEEL = 0x0800, /\* wheel button rolled \*/*

*MOUSEEVENTF\_MOVE\_NOCOALESCE = 0x2000, /\* The WM\_MOUSEMOVE messages will not be coalesced. The default behavior is to coalesce WM\_MOUSEMOVE messages. \*/*

*MOUSEEVENTF\_VIRTUALDESK = 0x4000, /\* map to entire virtual desktop \*/*

*MOUSEEVENTF\_ABSOLUTE = 0x8000, /\* absolute move \*/*

*}*

*[StructLayout(LayoutKind.Sequential)]*

*struct MouseInput*

*{*

*public int type;*

*public int dx;*

*public int dy;*

*public int mouseData;*

*public int dwFlags;*

*public int time;*

*public IntPtr dwExtraInfo;*

*};*

*[StructLayout(LayoutKind.Sequential)]*

*struct MouseMovePoint*

*{*

*public int x;*

*public int y;*

*public int time;*

*public IntPtr dwExtraInfo;*

*};*

*[DllImport("User32", EntryPoint = "SendInput")]*

*static extern uint SendInput(uint nInputs, IntPtr pInputs, int cbSize);*

*// GetSystemMetrics*

*public const int SM\_CXMAXTRACK = 59;*

*public const int SM\_CYMAXTRACK = 60;*

*public const int SM\_XVIRTUALSCREEN = 76;*

*public const int SM\_YVIRTUALSCREEN = 77;*

*public const int SM\_CXVIRTUALSCREEN = 78;*

*public const int SM\_CYVIRTUALSCREEN = 79;*

*public const int SM\_SWAPBUTTON = 23;*

*public const int SM\_CXSCREEN = 0;*

*public const int SM\_CYSCREEN = 1;*

*[DllImport("user32.dll")]*

*public static extern int GetSystemMetrics(int metric);*

*[DllImport("user32.dll")]*

*public static extern int GetDoubleClickTime();*

*internal static void AvoidDoubleClick()*

*{*

*int sleep = GetDoubleClickTime();*

*Thread.Sleep(sleep \* 2);*

*}*

*}*

*} [9]*

După cum se vede codul de la Microsoft respectă toate regulile generale. Mai departe acest cod va fi comparat cu cel al unui începător care a scris un notepad simplu.

*using System;*

*using System.Collections.Generic;*

*using System.ComponentModel;*

*using System.Data;*

*using System.Drawing;*

*using System.IO;*

*using System.Linq;*

*using System.Text;*

*using System.Threading.Tasks;*

*using System.Windows.Forms;*

*namespace Notepad*

*{*

*public partial class Form1 : Form*

*{*

*String path = String.Empty;*

*public Form1() => InitializeComponent();*

*private string ReturnMessageFromFormat(string type)*

*{*

*switch (type)*

*{*

*case "ino":*

*return "Arduino";*

*break;*

*case "cs":*

*return "C#";*

*break;*

*case "cpp":*

*return "C++";*

*break;*

*case "c":*

*return "C";*

*break;*

*case "btwo":*

*return "Braintwo";*

*break;*

*case "json":*

*return "Json";*

*break;*

*case "xml":*

*return "Xml";*

*break;*

*case "html":*

*return "HTML";*

*break;*

*case "css":*

*return "CSS";*

*break;*

*case "js":*

*return "JavaScript";*

*break;*

*default:*

*return "Text";*

*break;*

*}*

*}*

*private void openToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (openFileDialog1.ShowDialog() == DialogResult.OK)*

*{*

*textBox1.Text = File.ReadAllText(path = openFileDialog1.FileName);*

*string[] SplitExtension = openFileDialog1.FileName.Split('.');*

*labelFormat.Text = ReturnMessageFromFormat(SplitExtension[1]);*

*}*

*}*

*private void saveAsToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (saveFileDialog1.ShowDialog() == DialogResult.OK)*

*{*

*File.WriteAllText(path = saveFileDialog1.FileName, textBox1.Text);*

*}*

*}*

*private void saveToolStripMenuItem\_Click(object sender, EventArgs e) {*

*if (!String.IsNullOrWhiteSpace(path))*

*{*

*File.WriteAllText(path, textBox1.Text);*

*} else*

*{*

*saveAsToolStripMenuItem\_Click(sender, e);*

*}*

*}*

*private void exitPrompt()*

*{ DialogResult = MessageBox.Show("Do you want to save current file?",*

*"Notepad",*

*MessageBoxButtons.YesNoCancel,*

*MessageBoxIcon.Question,*

*MessageBoxDefaultButton.Button2); }*

*private void newToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (!String.IsNullOrWhiteSpace(textBox1.Text))*

*{*

*exitPrompt();*

*if (DialogResult == DialogResult.Yes)*

*{*

*saveToolStripMenuItem\_Click(sender, e);*

*textBox1.Text = String.Empty;*

*path = String.Empty;;*

*}*

*else if (DialogResult == DialogResult.No)*

*{*

*textBox1.Text = String.Empty;;*

*path = String.Empty;; }*

*} }*

*private void selectAllToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.SelectAll();*

*private void cutToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.Cut();*

*private void copyToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.Copy();*

*private void pasteToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.Paste();*

*private void deleteToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.SelectedText = String.Empty;*

*private void wordWrapToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (wordWrapToolStripMenuItem.Checked == true)*

*{*

*textBox1.WordWrap = false;*

*textBox1.ScrollBars = ScrollBars.Both;*

*wordWrapToolStripMenuItem.Checked = false;*

*}*

*else*

*{ textBox1.WordWrap = true;*

*textBox1.ScrollBars = ScrollBars.Vertical;*

*wordWrapToolStripMenuItem.Checked = true;*

*}*

*}*

*private void fontToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (fontDialog1.ShowDialog() == DialogResult.OK)*

*{*

*textBox1.Font = textBox1.Font = new Font(fontDialog1.Font, fontDialog1.Font.Style);*

*textBox1.ForeColor = fontDialog1.Color;*

*}*

*}*

*private void aboutToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*Form aboutForm = new Form2();*

*aboutForm.ShowDialog();*

*}*

*private void exitToolStripMenuItem\_Click(object sender, EventArgs e) => Application.Exit();*

*private void Form1\_FormClosing(object sender, FormClosingEventArgs e)*

*{*

*if (!string.IsNullOrWhiteSpace(textBox1.Text))*

*{*

*exitPrompt();*

*if (DialogResult == DialogResult.Yes)*

*{ saveToolStripMenuItem\_Click(sender, e);*

*}*

*else if (DialogResult == DialogResult.Cancel)*

*{ e.Cancel = true;*

*}*

*}*

*}*

*private void textBox1\_KeyDown(object sender, KeyEventArgs e)*

*{*

*if (e.Control)*

*{*

*switch (e.KeyCode)*

*{*

*case Keys.A:*

*e.SuppressKeyPress = true;*

*textBox1.SelectAll();*

*break;*

*case Keys.N:*

*e.SuppressKeyPress = true;*

*newToolStripMenuItem\_Click(sender, e);*

*break;*

*case Keys.S:*

*e.SuppressKeyPress = true; saveToolStripMenuItem\_Click(sender, e); break; } } }*

*private void blackToolStripMenuItem\_Click(object sender, EventArgs e)*

*{ textBox1.ForeColor = Color.White;*

*textBox1.BackColor = Color.Black;*

*this.BackColor = Color.Gray;*

*} private void grayToolStripMenuItem\_Click(object sender, EventArgs e)*

*{ textBox1.ForeColor = Color.Black;*

*textBox1.BackColor = Color.Gray;*

*this.BackColor = Color.Gray; }*

*private void defaultToolStripMenuItem\_Click(object sender, EventArgs e)*

*{ textBox1.ForeColor = Color.Black;*

*textBox1.BackColor = Color.White;*

*this.BackColor = Color.White; }*

*}} [10]*

Aici se observă că se respectă regula generală de păstrare a limitei maxime linii de cod, însă indentarea, filele și spațiile libere nu sunt peste tot folosite corect.

## Aplicarea instrumentelor pentru înlăturarea erorilor

Pentru aplicarea extensiilor se folosesc diferite scurtături, în Visual Studio aceasta este Ctrl+K + Ctrl+E, iar în redactorul textual VS code este Ctrl + K + D. Aceasta este necesar dacă nu este activată opțiunea format on save, însă dacă ea este activată la salvarea fișierului codul va fi formatat după opțiunile setate de utilizator.

Folosind instrumentul dat de Visual Studio a fost corectat codul începătorului. El trebuie să arate în modul următor:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Calculator

{

public partial class Form1 : Form

{

Double Result\_Value = 0;

String Operator\_Performed = " ";

bool PerformedOp = false;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button15\_Click(object sender, EventArgs e)

{

// numbers button and point

if (textBox\_Result.Text == "0" || PerformedOp)

textBox\_Result.Clear();

PerformedOp = false;

Button button = (Button)sender;

if(button.Text == ".")

{

if(!textBox\_Result.Text.Contains("."))

textBox\_Result.Text += button.Text;

}

else

textBox\_Result.Text += button.Text;

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void Operator\_click\_Event(object sender, EventArgs e)

{

// +, -, \*, / operators

Button button = (Button)sender;

if (Result\_Value != 0)

{

button16.PerformClick();

Operator\_Performed = button.Text;

label\_Show\_Op.Text = Result\_Value + " " + Operator\_Performed;

PerformedOp = true;

}

else

{

Operator\_Performed = button.Text;

Result\_Value = Double.Parse(textBox\_Result.Text);

label\_Show\_Op.Text = Result\_Value + " " + Operator\_Performed;

PerformedOp = true;

}

}

private void button5\_Click(object sender, EventArgs e)

{

//CLEAR ENTRY BUTTON

textBox\_Result.Text = "0";

}

private void button6\_Click(object sender, EventArgs e)

{

//CLEAR BUTTON

textBox\_Result.Text = "0";

Result\_Value = 0;

label\_Show\_Op.Text = " ";

}

private void button16\_Click(object sender, EventArgs e)

{

// EQUALS BUTTON

switch (Operator\_Performed)

{

case "+":

textBox\_Result.Text = (Result\_Value + Double.Parse(textBox\_Result.Text)).ToString();

break;

case "-":

textBox\_Result.Text = (Result\_Value - Double.Parse(textBox\_Result.Text)).ToString();

break;

case "\*":

textBox\_Result.Text = (Result\_Value \* Double.Parse(textBox\_Result.Text)).ToString();

break;

case "/":

textBox\_Result.Text = (Result\_Value / Double.Parse(textBox\_Result.Text)).ToString();

break;

default:

break;

}

Result\_Value = Double.Parse(textBox\_Result.Text);

label\_Show\_Op.Text = " ";

}

}

}

În acest mod trebuie să arate codul începătorului pentru calculatorul său. La fel a fost făcut codul mai citeț pentru codul aplicației de notepad.

*using System;*

*using System.Collections.Generic;*

*using System.ComponentModel;*

*using System.Data;*

*using System.Drawing;*

*using System.IO;*

*using System.Linq;*

*using System.Text;*

*using System.Threading.Tasks;*

*using System.Windows.Forms;*

*namespace Notepad*

*{*

*public partial class Form1 : Form*

*{*

*String path = String.Empty;*

*public Form1() => InitializeComponent();*

*private string ReturnMessageFromFormat(string type)*

*{*

*switch (type)*

*{*

*case "ino":*

*return "Arduino";*

*break;*

*case "cs":*

*return "C#";*

*break;*

*case "cpp":*

*return "C++";*

*break;*

*case "c":*

*return "C";*

*break;*

*case "btwo":*

*return "Braintwo";*

*break;*

*case "json":*

*return "Json";*

*break;*

*case "xml":*

*return "Xml";*

*break;*

*case "html":*

*return "HTML";*

*break;*

*case "css":*

*return "CSS";*

*break;*

*case "js":*

*return "JavaScript";*

*break;*

*default:*

*return "Text";*

*break;*

*}*

*}*

*private void openToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (openFileDialog1.ShowDialog() == DialogResult.OK)*

*{*

*textBox1.Text = File.ReadAllText(path = openFileDialog1.FileName);*

*string[] SplitExtension = openFileDialog1.FileName.Split('.');*

*labelFormat.Text = ReturnMessageFromFormat(SplitExtension[1]);*

*}*

*}*

*private void saveAsToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (saveFileDialog1.ShowDialog() == DialogResult.OK)*

*{*

*File.WriteAllText(path = saveFileDialog1.FileName, textBox1.Text);*

*}*

*}*

*private void saveToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (!String.IsNullOrWhiteSpace(path))*

*{*

*File.WriteAllText(path, textBox1.Text);*

*}*

*else*

*{*

*saveAsToolStripMenuItem\_Click(sender, e);*

*}*

*}*

*private void exitPrompt()*

*{*

*DialogResult = MessageBox.Show("Do you want to save current file?",*

*"Notepad",*

*MessageBoxButtons.YesNoCancel,*

*MessageBoxIcon.Question,*

*MessageBoxDefaultButton.Button2);*

*}*

*private void newToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (!String.IsNullOrWhiteSpace(textBox1.Text))*

*{*

*exitPrompt();*

*if (DialogResult == DialogResult.Yes)*

*{*

*saveToolStripMenuItem\_Click(sender, e);*

*textBox1.Text = String.Empty;*

*path = String.Empty;;*

*}*

*else if (DialogResult == DialogResult.No)*

*{*

*textBox1.Text = String.Empty;;*

*path = String.Empty;;*

*}*

*}*

*}*

*private void selectAllToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.SelectAll();*

*private void cutToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.Cut();*

*private void copyToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.Copy();*

*private void pasteToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.Paste();*

*private void deleteToolStripMenuItem\_Click(object sender, EventArgs e) => textBox1.SelectedText = String.Empty;*

*private void wordWrapToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (wordWrapToolStripMenuItem.Checked == true)*

*{*

*textBox1.WordWrap = false;*

*textBox1.ScrollBars = ScrollBars.Both;*

*wordWrapToolStripMenuItem.Checked = false;*

*}*

*else*

*{*

*textBox1.WordWrap = true;*

*textBox1.ScrollBars = ScrollBars.Vertical;*

*wordWrapToolStripMenuItem.Checked = true;*

*}*

*}*

*private void fontToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*if (fontDialog1.ShowDialog() == DialogResult.OK)*

*{*

*textBox1.Font = textBox1.Font = new Font(fontDialog1.Font, fontDialog1.Font.Style);*

*textBox1.ForeColor = fontDialog1.Color;*

*}*

*}*

*private void aboutToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*Form aboutForm = new Form2();*

*aboutForm.ShowDialog();*

*}*

*private void exitToolStripMenuItem\_Click(object sender, EventArgs e) => Application.Exit();*

*private void Form1\_FormClosing(object sender, FormClosingEventArgs e)*

*{*

*if (!string.IsNullOrWhiteSpace(textBox1.Text))*

*{*

*exitPrompt();*

*if (DialogResult == DialogResult.Yes)*

*{*

*saveToolStripMenuItem\_Click(sender, e);*

*}*

*else if (DialogResult == DialogResult.Cancel)*

*{*

*e.Cancel = true;*

*}*

*}*

*}*

*private void textBox1\_KeyDown(object sender, KeyEventArgs e)*

*{*

*if (e.Control)*

*{*

*switch (e.KeyCode)*

*{*

*case Keys.A:*

*e.SuppressKeyPress = true;*

*textBox1.SelectAll();*

*break;*

*case Keys.N:*

*e.SuppressKeyPress = true;*

*newToolStripMenuItem\_Click(sender, e);*

*break;*

*case Keys.S:*

*e.SuppressKeyPress = true;*

*saveToolStripMenuItem\_Click(sender, e);*

*break;*

*}*

*}*

*}*

*private void blackToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*textBox1.ForeColor = Color.White;*

*textBox1.BackColor = Color.Black;*

*this.BackColor = Color.Gray;*

*}*

*private void grayToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*textBox1.ForeColor = Color.Black;*

*textBox1.BackColor = Color.Gray;*

*this.BackColor = Color.Gray;*

*}*

*private void defaultToolStripMenuItem\_Click(object sender, EventArgs e)*

*{*

*textBox1.ForeColor = Color.Black;*

*textBox1.BackColor = Color.White;*

*this.BackColor = Color.White;*

*}*

*}*

*}*

În acest mod trebuie să arate codul începătorului.

## Elaborarea unei extensii pentru stilizarea codului

Extensiile sunt suplimente care permit de a personaliza și îmbunătăți experiența într-un IDE sau redactor textual prin adăugarea de noi funcții sau prin integrarea instrumentelor existente. O extensie poate varia la toate nivelurile de complexitate, dar scopul ei principal este de a crește productivitatea și de a satisface fluxul de lucru [11].

Pentru a crea o extensie la început trebuie ca programatorul să hotărască dacă va folosi un IDE sau redactor textual. Dacă se alege IDE-ul Visual Studio atunci va fi nevoie de a descărca Visual Studio SDK. Aici vor fi date câteva șabloane după care programatorul se poate orienta. După ce extensia e finisată aceasta poate fi publicată pe Visual Studio Marketplace unde mai mulți utilizatori o vor putea folosi sau se poate instala sau transmite fișierul cu extensia .vsx. [12]

După o analiză a mai multor surse despre cum trebuie să arate codul în limbajul C#, este de înțeles faptul că e dificil de a face o extensie pentru a formata codul, deoarece fiecare companie are preferințele proprii spre exemplu compania Google preferă să folosească o listă la folosirea variabilelor publice pe când compania Microsoft nu are așa preferință. De aceea pentru a face un instrument cât mai eficient trebuie de folosit regulile generale de stilizare pentru cod pentru a face extensia cât mai universală [13, 14, 15].

# CONCLUZII

In rezultatul studiului efectuat se poate de constatat că este necesar de a avea un cod style unit. Standardele de codificare sunt o serie de proceduri care pot fi definite pentru un anumit limbaj de programare, specificând un stil de programare, metode și proceduri diferite. Aceste proceduri pot fi pentru diferite aspecte ale programului scris în limbajul respectiv. Ele pot fi considerate atribute esențiale ale dezvoltării unui program.

Standardizarea are un impact pozitiv asupra oricărei afaceri. Și în industria software-ului, există anumite standarde de codare care sunt necesare pentru dezvoltarea unui software cu succes. Pentru majoritatea organizațiilor, funcționarea fără întreruperi a programelor software este esențială pentru creșterea lor.

Crearea de software și cod de bună calitate este un proces dificil. Este nevoie de eforturi consecvente din partea echipei de dezvoltare software pentru a îndeplini obiectivele de calitate a codului și pentru a urma cele mai bune practici de programare în echipă pentru a obține succes. Acesta este un lucru esențial pentru orice proiect software. Dezvoltatorii adesea ocolesc standardele de calitate atunci când li se cere să își îndeplinească sarcinile într-un interval scurt de timp.

Un standard de codare se asigură că toți dezvoltatorii care lucrează la proiect urmează anumite linii directoare specificate. Codul poate fi ușor de înțeles și este menținută consistența.

Consecvența are un impact pozitiv asupra calității programului și ar trebui să fie menținută în timpul codificării. De asemenea, trebuie de avut grijă ca regulile de codificare să fie urmate în mod omogen la diferite niveluri ale sistemului și nu să se contrazică reciproc. Codul programului terminat ar trebui să arate de parcă a fost scris de un singur dezvoltator, într-o singură sesiune.

Din această cauză e necesar de a avea un instrument care poate ajuta programatorul să scrie un cod cât mai rapid eficient și optimizat, întrucât aceasta va economisi foarte mult timp. Pentru aceasta este nevoie de a studia regulile generale ale unui limbaj, întrucât fiecare companie are diferite preferințe. Indiferent de limbaj, a decide cum ar trebui să arate codul este o parte importantă a calității generale a codului.

La fel trebuie de ales cu atenție unde va fi elaborată extensia într-un IDE sau redactor textual, deoarece diferența dintre acestea sunt foarte mari. Redactorul textual este mult mai simplu și poate merge pe un calculator mai puțin performat pe când IDE-ul conține și o interfață grafică pentru programa care va scrisă și poate folosi mai multe limbaje de programare.

# BIBLIOGRAFIE

1. *Wikipedia.org Programming Style* [online] [accesat 8 aprilie 2022]. Disponibil:

<https://en.wikipedia.org/wiki/Programming_style>

1. *Smashingmagazine.com de ce contează code style-ul* [online] [accesat 8 aprilie 2022]. Disponibil:

[https://www.smashingmagazine.com/2012/10/why-coding-style-matters/](https://www.smashingmagazine.com/2012/10/why-coding-style-matters/%20)

1. *MALINA Tran Reguli generale pentru formatarea codului tău* [online] [accesat 8 aprilie 2022]. Disponibil:

[https://medium.com/tech-and-the-city/10-rules-for-formatting-your-code-1153e60b28cf](https://medium.com/tech-and-the-city/10-rules-for-formatting-your-code-1153e60b28cf%20)

1. *Prettier.io site-ul oficial pentru extensia Prettier* [online] [accesat 12 aprilie 2022]. Disponibil: <https://prettier.io/>
2. *Eslint.org site-ul oficial pentru extensia ESLint* [online] [accesat 13 aprilie 2022]. Disponibil: <https://eslint.org/>
3. *Elnear.academy Diferența dintre ESLint și Prettier* [online] [accesat 8 aprilie 2022]. Disponibil: <https://enlear.academy/eslint-vs-prettier-57882d0fec1d>
4. *Codul companiei Microsoft care a scris calculatorul* [online] [accesat 23 aprilie 2022]. Disponibil: <https://github.com/microsoft/calculator>
5. *Codul începătorului care a scris calculatorul* [online] [accesat 20 aprilie 2022]. Disponibil: <https://github.com/petrediana/Calculator>
6. *Codul companiei Microsoft care a scris notepad-ul* [online] [accesat 25 aprilie 2022]. Disponibil: [https://github.com/microsoft/XmlNotepad](https://github.com/microsoft/XmlNotepad%20)
7. *Codul începătorului care a scris notepad-ul* [online] [accesat 20 aprilie 2022]. Disponibil: <https://github.com/Lavertis/simple-notepad>
8. *Ce este o extensie* [online] [accesat 12 aprilie 2022]. Disponibil:

<https://visualstudio.microsoft.com/vs/features/extend/>

1. *Microsoft guide pentru crearea extensiilor în IDE-ul Visual Studio* [online] [accesat 20 aprilie 2022]. Disponibil:

<https://docs.microsoft.com/en-us/visualstudio/extensibility/starting-to-develop-visual-studio-extensions?view=vs-2022>

1. *Microsoft style guide pentru C#* [online] [accesat 15 aprilie 2022]. Disponibil:

<https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>

1. *C# code style de la Google* [online] [accesat 17 aprilie 2022]. Disponibil:

<https://google.github.io/styleguide/csharp-style.html>

1. *Reguli generale de stilizare pentru limbajul C#* [online] [accesat 17 aprilie 2022]. Disponibil: <https://www.dofactory.com/csharp-coding-standards>