
Onni Lukkarila

# Ohjelmointikielen migraatio VB.NET:stä C#:iin suuressa liiketoimintakriittisessä järjestelmässä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja Viestintätekniikka

Insinöörityö

22.4.2025

# Tiivistelmä


| | |
|---|---|
| Tekijä: | Onni Lukkarila |
| Otsikko: | Ohjelmointikielen migraatio VB.NET:stä C#:iin suuressa liiketoimintakriittisessä järjestelmässä |
| Sivumäärä: | 32 sivua |
| Aika: | 22.4.2025 |
| Tutkinto: | Insinööri (AMK) |
| Tutkinto-ohjelma: | Tieto- ja Viestintätekniikka |
| Ammatillinen pääaine: | Ohjelmistotuotanto |
| Ohjaajat: | Lehtori Jormä Räty |


---


Opinnäytetyön tarkoitus on tutkia Visual Basic.NET -ohjelmointikielen muutosta nykyaikaisempaan C#:iin suuren liikentoimintakriittisen järjestelmän sisällä. Työn tavoitteena on löytää syitä migraation hyödyllisyydelle sekä havaita mahdollisia haasteita migraatioon liittyen.

Opinnäytetyön aikana selvitin VB.NET- ja C#-ohjelmointikielten eroja erityisesti teknisestä näkökulmasta. Tein työssä omia suorituskykytestejä sekä syntaksivertailuja kielten välillä. Testasin myös CodeConverter-nimisen konversiotyökalun käytettävyyttä kehityksen aikana. Teknisten eroavaisuuksien lisäksi pohdin myös ohjelmointikielen migraatiota organisaation kannalta.

Opinnäytetyö on toteutettu yhdessä Verohallinnon kanssa. Verohallinto korvasi kymmenittäin räätälöityjä verolajikohtaisia tietojärjestelmiä uudella GenTax-valmisohjelmistolla, joka on FAST-yrityksen luoma. Ohjelmointikielen migraatio tehdään GenTax-järjestelmään lähivuosien aikana.



Avainsanat: VB.NET, C#, Migraatio


---

## Abstract


| | |
|---|---|
| Author: | Onni Lukkarila |
| Title: | Programming Language Migration from Visual Basic to C# In Large Business-Critical System |
| Number of Pages: | 32 pages |
| Date: | 22 April 2025 |
| Degree: | Bachelor of Engineering |
| Degree Programme: | Information and Communication Technology |
| Professional Major: | Software Engineering |
| Supervisors: | Jorma Räty, Senior Lecturer |


---


The purpose of this thesis is to introduce the transition from Visual Basic.NET programming language to the more modern C# within a large business-critical system. The goal is to present reasons for the usefulness of this migration, as well as to identify possible challenges related to the migration.

During the study, the differences between VB.NET and C# programming languages was investigated, particularly from a technical perspective. Performance tests were conducted, and the syntaxes of both languages were compared. Also, the usability of a conversion tool called CodeConverter was tested during development. In addition to technical differences, the programming language migration from an organizational perspective was considered, too.

The study was carried out in collaboration with the Finnish Tax Administration. The Tax Administration replaced dozens of customized tax-specific information systems with a new GenTax off-the-shelf software created by FAST company. The programming language migration will be done to the GenTax system in the coming years.



Keywords: VB.NET, C#, Migration

# Sisällys

## Lyhenteet

VB.NET: *Visual Basic .NET.* Microsoftin kehittämä ohjelmointikieli, joka perustuu 60-luvulla luotuun BASIC-ohjelmointikieleen.

C# Microsoftin kehittämä, moderni, oliopohjainen ohjelmointikieli, joka julkaistiin .NET-ohjelmistokehyksen kanssa vuonna 2002.

BASIC *Beginner's All-purpose Symbolic Instruction Code.* 1960-luvulla kehitetty ohjelmointikieli, johon Visual Basic perustuu.

.NET Microsoftin kehittämä ohjelmistokehys, joka mahdollistaa sovellusten kehittämisen ja suorittamisen eri alustoilla.

CLR *Common Language Runtime.* .NET-ohjelmistokehyksen ydinkomponentti, joka hallitsee .NET-ohjelmien ajamisen ja tarjoaa niille keskeiset toiminnot.

FCL *Framework Class Library.* .NET-ohjelmistokehyksen kokonaisvaltainen oliopohjainen luokkakirjasto, joka sisältää luokkia ja rajapintoja kehittäjien käyttöön.

CIL *Common Intermediate Language. V*älikieli, johon .NET-ohjelmat käännetään ennen konekieleksi kääntämistä.

I/O *Input/Output.* Viittaa tiedonsiirtoon tietokoneen komponenttien tai ohjelmien välillä.

# 1 Johdanto

Ohjelmistokielen vaihdosta suunnitellessa tulee ottaa huomioon monia erilaisia tekijöitä. Varsinkin liiketoiminnallisesti kriittisessä järjestelmässä täytyy huomioida kaikki mahdolliset hyödyt sekä haitat huomioon jo migraatiota suunnitellessa. Tämän opinnäytetyön tavoite on tutkia Visual Basic.NET -ohjelmointikielen muutosta nykyaikaisempaan C#:iin suuren järjestelmän sisällä. Tulen työssäni tutkimaan, mitä hyötyjä ohjelmointikielen vaihtamisesta on etenkin teknisestä näkökulmasta katsottuna, mutta myös yleisesti organisaation kannalta. Tavoitteenani on löytää konkreettisia hyötyjä migraatiosta, jotka motivoivat sen tekemistä. Tulen myös tutkimaan migraatiota varten ratkaistavia ongelmia ja niille mahdollisia ratkaisuja. Tavoitteenani on selvittää kaikille isoimmille ongelmille mahdollinen ratkaisu ja mitä se vaatii.

Järjestelmä, jota käsittelen opinnäytetyön aikana, on Verohallinnon käyttämä valmisohjelmisto GenTax. Se on amerikkalaisen FAST-yrityksen tarjoama valmisohjelmistopohjainen järjestelmä, jolla Verohallinto korvasi kymmenittäin räätälöityjä verolajikohtaisia tietojärjestelmiä vuosien 2014–2019 aikana (Verohallinto 2019). Järjestelmän keskeisyys ja kriittisyys Suomen verotuksen toteuttamisessa onkin varsin ilmiselvää.

Ohjelmistojärjestelmien elinkaaren aikana niiden ylläpidettävyys ja laajennettavuus nousevat kriittisiksi tekijöiksi järjestelmän arvon säilyttämisessä. Varsinkin julkishallinnon järjestelmissä, kuten Verohallinnon käyttämissä ohjelmistoissa. Teknologiavalintojen merkitys korostuu järjestelmien pitkien elinkaarien vuoksi. Ohjelmointikielen vaihtaminen edustaa merkittävää strategista päätöstä, joka vaikuttaa järjestelmän tulevaisuuden kehitysmahdollisuuksiin, ylläpidettävyyteen sekä organisaation kykyyn rekrytoida osaavaa henkilöstöä. Tämän opinnäytetyön tarkoituksena on tarjota kattava analyysi VB.NET-kielen vaihtamisesta C#-kieleen, kun painotetaan erityisesti teknisiä näkökohtia ja mahdollisia suorituskykyeroja unohtamatta organisaation näkökulmaa.

Tulen opinnäytetyön aikana ensiksi esittelemään ohjelmointikielten sekä niihin vahvasti liittyvien teknologioiden historiaa, nykyhetkeä ja tulevaisuuden

näkymiä. Tämän jälkeen tutkin sekä esittelen kielten eroja niiden syntaksista lähtien, aina mahdollisiin suorituskykyeroihin asti. Tarkoitukseni on myös arvioida, miten nämä erot ja erojen suuruuksista johtuvat haasteet vertaantuvat muunlaisiin haasteisiin. Lopuksi käyn vielä läpi tutkimusten tulokset ja teen yhteenvedon työstäni.

Valitsin tutkittaviksi kieliksi juuri VB.NET:n ja C#:n siitä syystä, että opinnäytetyössä käsiteltävän järjestelmän tulevassa versiossa ohjelmointikieli tulee vaihtumaan VB.NET:stä C#:iin.

# 2 Lähtökohdat

## 2.1 Ohjelmiston nykytilanne

Opinnäytetyön aikana käsiteltävä ohjelmisto käyttää tällä hetkellä VB.NET-ohjelmointikieltä ja luonnollisesti .NET-ohjelmointikehystä. Järjestelmä käyttää .NET-ohjelmointikehyksen versiota 4.7. Pääasiallisena ohjelmointiympäristönä toimii Microsoftin kehittämä Visual Studio.

## 2.2 Miksi muutos halutaan

Syitä siihen, miksi muutos halutaan tehdä, on jo jonkin verran tiedossa. Ohjelmiston uusimmissa versioissa on käytössä pelkästään C#-kieli, joten luonnollisesti se luo motivaatiota migraatiolle. Verohallinto haluaa luonnollisesti pysyä mahdollisimman hyvin mukana ohjelmiston versiokehityksessä, jotta ohjelmistolla säilyisi mahdollisimman hyvä tuki ohjelmiston tarjoajalta. Yleisesti on ymmärretty, että ohjelmistojen uusimmissa versioissa on muun muassa parhaimmat tietoturvapäivitykset ja niihin tulee tarpeen vaatiessa nopeimmin päivitykset sekä korjaukset.

C# on myös tämänhetkisen käsityksen mukaan yksinkertaisesti suositumpi ohjelmointikieli. Jos käsitys on totta, tarkoittaisi se esimerkiksi isompaa ekosysteemiä kielen ympärillä. Tämä tarkoittaisi sitä, että esimerkiksi ongelmatilanteen syntyessä kehittämisen aikana, olisi huomattavasti todennäköisempää, että

ongelmaan voisi löytyä ratkaisu. Mitä suositumpi kieli on, sitä enemmän siihen löytyy myös muiden luomia opetusmateriaaleja. Kielen suosio on myös siis suuri tekijä, kun mietitään ohjelmointikielen etuja. Tulen tämän opinnäytetyön aikana selvittämään, onko olettamus C#:n suosiosta totta.

## 2.3 Tiedossa olevat haasteet

Suurin tiedossa oleva haaste on yksinkertaisesti nykyisen järjestelmän koko ja sen kriittisyys. Järjestelmän koko ja ohjelmakoodin määrä on erittäin suuri, joten ajallisesti kielen muutokseen tulee kulumaan hyvin paljon aikaa. Ohjelmointikielen vaihto tarkoittaa myös sitä, että koko järjestelmä ja sen toiminnallisuus pitää regressiotestata. Tämän kokoluokan järjestelmässä siihen tulee myös kulumaan paljon henkilötyövoimaa. Monien ominaisuuksien testaamiseen käytetään toki automaatiotestausta, mutta iso osa toiminnallisuuksista vaatii myös käyttäjätestausta. Verohallinnon GenTax-järjestelmän ylläpitotoimiin kuuluu jo säännöllinen automaatiotestaus, mikä varmasti auttaa näin suuren kehitysprojektin kanssa. Toisaalta automaatiotestauksen testitapauksia saatetaan joutua muokkaamaan ohjelmointikielen muutoksen myötä.

Verohallinnolla on käytössä vuosittainen niin sanottu ”release”-aikataulu. Tällä tarkoitetaan sitä, että joka vuosi GenTax-järjestelmään julkaistaan isompi päivitys, jonka yhteydessä järjestelmään julkaistaan paljon uusia toiminnallisuuksia. Vuosittaisin release-päivityksen lisäksi GenTax-järjestelmään toteutetaan myös lähes koko vuoden ajan jatkuvaa kehitystä. Jatkuvassa kehityksessä toteutetaan yleisesti muun muassa virheenkorjauksia ja luodaan hieman pienempiä uusia ominaisuuksia. Ohjelmointikielen migraatio tulee olemaan niin suuri päivitys, että se tullaan hyvin todennäköisesti toteuttamaan release-päivityksen yhteydessä. Migraation toteuttaminen release-päivityksessä on myös testauksen kannalta hyvä idea, sillä Verohallinon release-malliin kuuluu jo ennestään järjestelmän toimintojen suuri regressiotestaus ennen suurta päivitystä.

Kun niinkin merkittävä osa kehittäjien päivittäisistä työkaluista muutetaan kuin ohjelmointikieli, on hyvä miettiä tarvetta henkilöstön uudelleenkouluttamiselle. Vaikka kielet muistuttaisivatkin melko paljon toisiaan ja suurin osa

toiminnallisuuksista pystyttäisiin rakentamaan samalla tavalla, ei pidä aliarvioida kielen muutoksesta johtuvia muutoksia jokapäiväisessä työssä. Toki kokenut kehittäjä ei lähde nollasta, kun vaihdetaan VB.NET:sta C#:iin, koska molemmat kielet ovat Microsoftin luomia olio-ohjelmointikieliä, joten niistä löytyy paljon samanlaisuuksia.

VB.NET-kielellä luodun järjestelmän C#--kieliseksi muuntamiseen on myös olemassa erilaisia konversiotyökaluja. GenTax-järjestelmän koodissa käytetään kuitenkin paljon FAST:n kehittämiä omia kirjastoja, jotka todennäköisesti vaikeuttavat ulkopuolisten konvertointityökalujen käyttöä. FAST on jo aiemmin tehnyt samankaltaisia VB.NET -> C# konverioita, joita varten se on kehittänyt oman, Visual Studioon integroidun konversiotyökalun. Tulen tämän työn aikana myös tarkastelemaan kyseisen työkalun käyttöä.

# 3 Teknologiat

## 3.1 Visual Basic .NET

Visual Basic .NET perustuu 60-luvulla luotuun BASIC-ohjelmointikieleen. Vuonna 1991 Microsoft otti suuren harppauksen kehittämällä ensimmäisen version Visual Basic -kielestä. Visual Basic eli VB luotiin BASIC-kielen pohjalta uutta Windows-käyttöjärjestelmää varten. Kun Microsoft alkoi kehittää uutta .NET-ohjelmistokehystä, haluttiin myös VB tuoda osaksi heti sen ensimmäistä versiota. Näinpä vuonna 2002 julkaistiin uusi VB.NET-kieli. Harppaus VB:stä VB:NET:iin merkitsi suurta muutosta VB-ohjelmoijille, sillä päivityksen seurauksena kieli siirtyi olio-ohjelmoinnin mukaiseksi. (Vick 2004.)

Vaikka VB.NET-kielen historia on hyvin pitkä ja nykypäivänä koettu hieman vanhentuneeksi, Microsoft aikoo edelleen jatkaa sen uusimpien versioiden tukemista. .NET-ohjelmointikehyksen ydinkirjastot tukevat kieltä tulevaisuudessakin, ja myös monet tulevaisuuden parannukset .NET- ohjelmointikehyksen suorituskykyyn ja kirjastoihin tulevat automaattisesti parantamaan myös VB.NET:iä. (Microsoft 2023.)

## 3.2 C#

Alkuperäisesti vuonna 2002 julkaistu C#-ohjelmointikielen versio 1.0 julkaistiin .NET-ohjelmointikehyksen kanssa. ECMA Internationalin (Ecma International 2001) asettamien suunnittelutavoitteiden mukaan C# pyrki olemaan yksinkertainen, moderni ja yleiskäyttöinen olio-ohjelmoinnin käytäntöihin perustuva ohjelmointikieli. Ensimmäisessä versiossaan kieli muistuttikin hyvin paljon Java-ohjelmointikieltä, mikä toisaalta tarkoitti, että se oli saavuttanut sille asetetut tavoitteet. (Microsoft 2024b.)

Nykypäivänä C# on monipuolinen kieli, jota käytetään erityisesti Windows-kehitykseen, mutta myös laajasti sen ulkopuolella. Kieli tukee alustariippumatonta kehitystä .NET Coren ansiosta, ja näin sen käytön verkkosovelluksissa, pilvipalveluissa, mobiilisovelluksissa sekä pelikehityksessä. Nämä ovat muutamia syitä, joiden ansiosta C# on vakiinnuttanut itsensä keskeiseen rooliin yritysohjelmistojen, pelien ja muiden sovellusten kehittämisessä.

## 3.3 .NET-ohjelmistokehys

Microsoft .NET Framework on ohjelmistokehys, joka mahdollistaa .NET-ohjelmien suorittamisen. Se toimii pohjana, jonka päällä .NET-ohjelmat toimivat. .NET-ohjelmistokehys tukee hajautettuja ympäristöjä, joissa laskentatehtävät voidaan suorittaa palvelimilla Internetin kautta, mutta myös perinteiset työpöytäsovellukset toimivat sen avulla. (Moghadampour 2009.)

### 3.3.1 .NET Frameworkin evoluutio

Microsoft on vuosien varrella kehittänyt .NET-ohjelmistokehystä vastaamaan ohjelmistokehityksen muuttuvia tarpeita. Alun perin vuonna 2002 julkaistu .NET Framework oli suunniteltu ensisijaisesti Windows-ympäristöön, mikä rajoitti sen käyttöä muilla alustoilla. (Microsoft 2023b.) Tämä alustariippuvuus alkoi muodostua rajoitteeksi pilvipalveluiden ja mobiilisovellusten yleistyessä.

Vastatakseen näihin haasteisiin Microsoft julkaisi vuonna 2016 .NET Coren, joka oli täysin uudelleenkirjoitettu, avoimen lähdekoodin alustariippumaton versio .NET-ohjelmistokehyksestä (Hunter, 2019). .NET Core mahdollisti sovellusten kehittämisen ja suorittamisen Windows-, Linux- ja macOS-alustoilla. Se toi mukanaan myös merkittäviä suorituskykyparannuksia ja moderneja ominaisuuksia, kuten paremman tuen mikropalveluarkkitehtuurille ja pilvipohjaisille sovelluksille. (Microsoft 2024a.)

Vuonna 2020 Microsoft otti merkittävän askeleen .NET-ekosysteemin yhtenäistämisessä julkaisemalla .NET 5:n (Lander, 2019). Tämä versio yhdisti .NET Frameworkin ja .NET Coren parhaat puolet yhteen yhtenäiseen alustaan, jota kutsutaan nykyään yksinkertaisesti nimellä ".NET". Versionumerointi jatkui suoraan .NET Coresta, mutta "Core"-nimitys pudotettiin pois. Myöhemmät versiot aina 2021 julkaistusta .NET 6:sta loppuvuodesta 2025 julkaistavaan .NET 10:een ovat jatkaneet tätä yhtenäistämistä ja tuoneet lisää ominaisuuksia .NET-kehittäjien käyttöön. (Microsoft 2024a.)

Tämä .NET-ohjelmistokehyksen evoluutio on merkityksellinen myös ohjelmointikielen valinnan kannalta. C# on saanut enemmän kehityspanosta ja uusia ominaisuuksia .NET 5:n ja sitä uudempien versioiden myötä, kun taas VB.NET on jäänyt enemmän ylläpitotilaan. Tulevaisuuden yhteensopivuuden näkökulmasta C# tarjoaa paremman tuen modernille .NET-kehitykselle, mikä on tärkeä huomioida migraatiota suunniteltaessa.

### 3.3.2 Common Language Runtime (CLR)

.NET ohjelmistokehyksen arkkitehtuuri koostuu kahdesta pääkomponentista. Toinen niistä on Common Language Runtime tai CLR. CLR on käytännössä .NET-ohjelmistokehyksen sydän. Se hallitsee .NET-ohjelmien ajamisen sekä tarjoaa ohjelmille niiden keskeiset toiminnot kuten muistin hallinnan, säikeiden hallinnan sekä roskan kerääjän (garbage collector). CLR varmistaa, että .NET-ohjelmat toimivat turvallisessa ja hallitussa ympäristössä, jossa erilaiset ongelmat kuten muistivuodot, pointtereiden väärinkäytöt sekä muut yleiset ohjelmointivirheet minimoidaan. (Cybellium 2023.)

CLR myös mahdollistaa eri ohjelmointikielillä kirjoitetun koodin yhteistoiminnan. Se on siis luonnollisesti myös yhdistävä tekijä VB.NET- ja C#-ohjelmointikielten välillä. Käytännössä CLR yhdistää kieliä .NET-ohjelman kääntämisvaiheessa. Ensiksi .NET-ohjelma käännetään kielestä riippumatta yleiseksi Common Intermediate Language (CIL) -kieleksi. Tämä CIL-kieli muodostaa niin sanotun assembly-tiedoston, jonka CLR kokoaa konekieleksi ennen ohjelman ajoa. (Cybellium 2023.)

### 3.3.3 Framework Class Library (FCL)

Toinen pääkomponentti, mistä .NET-ohjelmistokehys muodostuu, on Framework Class Library eli FCL. FCL on kokonaisvaltainen, oliopohjainen kokoelma, joka sisältää muun muassa luokkia ja rajapintoja, joita kehittäjät voivat käyttää kehittäessään .NET-ohjelmia.

FCL on jaettu niin sanottuihin nimiavaruuksiin (namespace), joista jokainen sisältää samaan kategoriaan liittyviä luokkia. Esimerkiksi ”System.IO”-nimiavaruus sisältää luokkia tiedostojen ja tietovirtojen käsittelyyn, kun taas ”System.Net”-nimiavaruus sisältää luokkia, joita käytetään verkkokehitykseen. (Cybellium 2023.)

## 3.4 Visual Studio

Visual Studio on Microsoftin luoma ohjelmointiympäristö, joka on suunniteltu varta vasten .NET-kehitystä varten. Ohjelmointiympäristö tarkoittaa ympäristöä, jossa voi kirjoittaa, muokata, testata ja ajaa ohjelmakoodia. Visual Studio sisältää muun muassa erilaisia koodin kokoajia, avustajia, lähteen hallintaa sekä laajennuksia, jotka mahdollistavat sekä parantavat kehityskokemusta. (Microsoft 2024c.)

Visual Studio tukee monia .NET-ohjelmointikieliä, mukaan lukien VB.NET ja C#. Se tarjoaa myös erilaisia projektimalleja, joiden pohjalta on helpompi aloittaa uuden .NET-projektin kehitys. Erilaisia projektimalleja on esimerkiksi konsolisovelluksille, verkkosovelluksille sekä Windows-sovelluksille.

### 3.5 Ohjelmointikoodin konvertointityökalu

GenTax-ohjelmiston tuottaja FAST on kehittänyt sisäiseen käyttöönsä ohjelmointikielen konvertointiin käytettävän työkalun. Työkalu on rakennettu CodeConverter-työkalua hyväksikäyttäen. CodeConverter on avoimen lähdekoodin työkalu, joka mahdollistaa koodin automaattisen konvertoinnin Visual Basic .NET -kielestä C#-kieleen ja päinvastoin. Työkalu on saatavilla GitHub-repositoriossa (CodeConverter, 2024), ja sen kehitys on edelleen aktiivista. CodeConverter-työkalun saa asennettua Visual Studion kauppapaikalta laajennuksena suoraan Visual Studioon, mutta sillä on myös nettisivu, jossa sitä voi käyttää koodin kääntämiseen. CodeConverter hyödyntää Roslyn-kääntäjäalustaa, joka mahdollistaa tarkan syntaksipuun (syntax tree) analyysin koodien välillä. (Microsoft 2024e.)

Työkalu konvertoi suuren osan VB.NET-koodista C#-kielelle, mutta kehittäjällä on silti paljon tehtävää konversiossa. Vaikka työkalu ei tuottaisi täydellistä lopputulosta kaikissa tapauksissa, se todennäköisesti säästää merkittävästi aikaa verrattuna manuaaliseen koodin uudelleenkirjoitukseen. Tulen opinnäytetyön toteutusluvussa tutustumaan konvertointityökalun käyttöön tarkemmin sekä havainnollistan sen käyttöä esimerkein.

## 4 Opinnäytetyön toteutus

### 4.1 Tekninen puoli

Opinnäytetyöni toteutuksen teknisessä osassa kiinnitän huomiota sekä kielten välisiin suorituskyvyllisiin eroihin että syntaksisiin eroihin. Tulen selvittämään, kuinka erilaisia kielet loppujen lopuksi ovat teknisellä tasolla. Esittelen kielten eroja visuaalisilla taulukoilla sekä testiohjelmilla, joilla testaan molempien kielien suorituskykyä.

Tulen myös esittelemään tarkemmin CodeConverter-työkalua, jota todennäköisesti tullaan käyttämään ohjelmointikielen migraatiossa kehittäjää auttavana työkaluna. Haluan etenkin saada selville, kuinka helppokäyttöinen työkalu on

sekä kuinka paljon se voi helpottaa kehittäjän tekemää konversiotyötä. Toisaalta pohdin myös, mitä kaikkea kehittäjän pitää muistaa tehdä ennen ja jälkeen konversiotyökalun käytön.

### 4.1.1 Syntaksiset erot

Vaikka C# ja VB.NET lopulta muotoutuvat samanlaiseksi CIL-kieleksi ennen ohjelman ajoa, ohjelmakoodin kirjoitusvaiheessa syntaksisia eroja näiden kahden kielen välillä on useita. VB.NET näyttää usein enemmän verbaalilta kieleltä verrattuna C#-kieleen. Tästä syystä se voi olla jopa helpompi kieli oppia aloittelevalle koodaajalle. Muutamia esimerkkejä tästä on seuraavassa taulukossa.

Taulukko 1. Kielten väliset syntaksiset erot

| Toiminto | VB.NET | C# |
|---|---|---|
| Metodi | `Public Sub MethodName()`<br>`    ...`<br>`End Sub` | `public void MethodName() {`<br>`    ...`<br>`}` |
| Ehtorakenne | `If x > 0 Then`<br>`    ...`<br>`End If` | `if (x > 0) {`<br>`    ...`<br>`}` |
| Muuttujan määrittely | `Dim x As Integer = 10` | `int x = 10;` |
| For-silmukka | `For i As Integer = 0 To 9`<br>`    ...`<br>`Next` | `for (int i = 0; i < 10; i++) {`<br>`    ...`<br>`}` |
| While-silmukka | `While x < 10`<br>`    ...`<br>`End While` | `while (x < 10) {`<br>`    ...`<br>`}` |
| Operaattorit | `And, Or, Not, AndAlso, OrElse` | `&&, \|\|, !` |

Kuten taulukosta 1 huomaa, VB.NET-kielen monet syntaksiset ominaisuudet ovat huomattavasti verbaalisempia kuin C#-kielen vastaavat ominaisuudet. Tästä syystä VB.NET saatta olla ensimmäistä ohjelmointikieltään opettelevalle hyvin luonnollinen kieli opeteltavaksi. Toisaalta C#-kielen syntaksi koetaan hieman modernimmaksi, joten jos ohjelmoijalla on jo kokemusta jostain toisesta modernimmasta kielestä, kokee hän todennäköisesti C#:in luontevammaksi.

Yksi pieni, mutta kuitenkin merkittävä ero kielten syntaksin välillä on se, että C# on ns. case-sensitive-kieli. Tämä tarkoittaa sitä, että C#-kielessä merkkien

kirjainkoolla on väliä, toisin kuin VB.NET-kielessä. (Mojica 2002.) Esimerkkikoodissa 1 on pieni esimerkki havainnollistamaan kirjainkoon merkittävyyttä.

```
‘VB.NET esimerkki
Dim MyVariable As Integer = 10
Dim myVariable As Integer = 20 'Kieli ei salli tätä; sama nimi.

//C# esimerkki
var MyVariable = 10;
var myVariable = 20; // Tämä on eri muuttuja kuin yllä.
```

Esimerkkikoodi 1. Kirjainkoon merkittävyyttä havainnollistava esimerkkikoodi VB-NET- ja C#-kielellä

### 4.1.2 Suorituskyvylliset erot

Puhdasta suorituskykyä vertailin opinnäytetyössäni tekemällä pienen testiohjelman, joka etsii niin sanottuja Munchausen-lukuja. Munchausen-luku on erityinen kokonaisluku, jossa jokainen numero korotettuna itsensä potenssiin summataan, ja lopputulos on alkuperäinen luku. Esimerkiksi luku 3435 on Munchausen-luku, koska $3^3 + 4^4 + 3^3 + 5^5 = 3435$. (Wolfram MathWorld 2025.)

Testiohjelma käy läpi kaikki luvut välillä 0–440 000 000 ja tarkistaa, täyttävätkö ne Munchausen-luvun ehdon. Optimointia varten ohjelma hyödyntää välimuistia (cache-taulukkoa), johon on esilaskettu jokaisen numeron $n^n$-arvot välillä 0–9. Tämä vähentää laskennan aikana tehtävien eksponenttifunktiokutsujen määrää, mikä nopeuttaa suoritusta. Ohjelman suoritusajan mittaamiseen käytin Stopwatch-luokkaa, joka tarjoaa tarkkaa ajanseurantaa. Seuraavana on molemmilla kielillä toteutetut ohjelmakoodit.

```
using System.Diagnostics;
using System;
class Program
{
    private const int LIMIT = 440_000_000;
    private int[] cache;

    public Program()
    {
        this.cache = this.GetCache();
    }
    private int[] GetCache()
    {
        int[] cache = new int[10];
        cache[0] = 0;
        for (int i = 1; i <= 9; ++i)
        {
            cache[i] = (int)Math.Pow(i, i);
        }
        return cache;
    }
    private bool IsMunchausen(int number)
    {
        int n = number;
        int total = 0;

        while (n > 0)
        {
            int digit = n % 10;
            total += cache[digit];
            if (total > number)
            {
                return false;
            }
            n = n / 10;
        }
        return total == number;
    }

    private void Start()
    {
        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start();

        for (int i = 0; i < LIMIT; ++i)
        {
            if (IsMunchausen(i))
            {
                Console.WriteLine(i);
            }
        }
        stopwatch.Stop();
        Console.WriteLine("Elapsed time: " + stopwatch.Elapsed);
    }

    public static void Main(string[] args)
    {
        Program p = new Program();
        p.Start();
    }
}
```

Kuva 1. Suorituskykytesti C#-kielellä

```vb
Imports System
Module VBModule
    Private Const LIMIT As Integer = 440000000
    Private cache() As Integer

    Sub Main()

        cache = GetCache()
        Start()

    End Sub

    Private Function GetCache() As Integer()
        Dim cache(9) As Integer
        cache(0) = 0
        For i As Integer = 1 To 9
            cache(i) = CInt(Math.Pow(i, i))
        Next
        Return cache
    End Function

    Private Function IsMunchausen(number As Integer) As Boolean
        Dim n As Integer = number
        Dim total As Integer = 0

        While n > 0
            Dim digit As Integer = n Mod 10
            total += cache(digit)
            If total > number Then
                Return False
            End If
            n \= 10
        End While

        Return total = number
    End Function

    Private Sub Start()
        Dim stopwatch As New Stopwatch
        stopwatch.Start()

        For i As Integer = 0 To LIMIT - 1
            If IsMunchausen(i) Then
                Console.WriteLine(i)
            End If
        Next

        stopwatch.Stop()
        Console.WriteLine($"Execution Time: {stopwatch.Elapsed}")
    End Sub
End Module
```

Kuva 2. Suorituskykytesti VB.NET-kielellä

Ajoin molemmat ohjelmat viisi kertaa ja jokaisella kerralla otin ylös ohjelman kokonaissuoritusajan. Ohjelmien ajot suoritettiin henkilökohtaisella tietokoneellani, jossa on Intel i7-7700K @ 4.20GHz -prosessori ja Windows-käyttöjärjestelmä.

Ohjelmien suoritus tapahtui Visual Studio 2022 -ohjelmointiympäristössä. Taulukossa 2 näkyvät molempien kielten suoritusajat.

Taulukko 2. Munchausen-ohjelman suoritusajat sekunteina

| **Ajo #** | **C#** | **VB.NET** |
|---|---|---|
| 1. | 23,941 | 25,165 |
| 2. | 23,956 | 25,412 |
| 3. | 23,900 | 25,277 |
| 4. | 23,825 | 24,780 |
| 5. | 23,891 | 24,911 |
| KA | 23,903 | 25,109 |

Kuten taulukon 2 testitulokset osoittavat, C# suoriutui testatuista tehtävistä keskimäärin noin 5 % nopeammin kuin VB.NET. Tämä todennäköisesti johtuu ensisijaisesti siitä, että C# on suunniteltu suorituskykyä silmällä pitäen, ja sen kääntäjä pystyy optimoimaan koodia tehokkaammin. VB.NET puolestaan tekee ajonaikaisia lisätarkistuksia, mikä voi vaikuttaa suoritusaikaan ja hidastaa ohjelman toimintaa joissakin tilanteissa. Molemmat kielet käännetään ajon aikana samankaltaiseksi CIL-koodiksi, mikä tasoittaa niiden suorituskykyeroja, mutta C#:n etuna on kuitenkin kyky tuottaa joissakin tapauksissa hieman optimoidumpaa CIL-koodia. Tämä voi näkyä parempana suorituskykynä. (Sathiaseelan & Sasikaladevi 2009). Tämä todennäköisesti selittää testissä havaitun pienen eron suoritusaikojen välillä.

Halusin myös puhtaan laskennallisen suorituskyvyn lisäksi testata kielten välisiä eroja asynkronisissa ominaisuuksissa. Loin tätäkin testausta varten pienen testiohjelman molemmilla kielillä, jossa suoritan rinnakkaisia asynkronisia operaatioita. Ohjelmat simuloivat hieman todellisempaa työkuormaa yhdistämällä sekä I/O-odotusta (Task.Delay) että puhdasta laskentatyötä. Ohjelma suorittaa

operaatiot kymmenen kertaa ja tämän jälkeen laskee keskiarvoisen suoritusajan. Ajoin nämä ohjelmat kuitenkin myös viisi kertaa molemmilla kielillä saadakseni mahdollisimman oikeat tulokset. Käytin suoritusajan mittaamiseen tälläkin kertaa Stopwatch-luokkaa. Seuraavana on ohjelman koodi molemmilla kielillä.

```
using System;
using System.Diagnostics;
using System.Threading.Tasks;

class Program
{
    static async Task Main()
    {
        int iterationCount = 10;
        double totalTime = 0;

        for (int i = 0; i < iterationCount; i++)
        {
            var elapsed = await MeasureParallelOperationsAsync();
            totalTime += elapsed;
            Console.WriteLine($"Lap {i + 1}: {elapsed} ms");
        }

        double averageTime = totalTime / iterationCount;
        Console.WriteLine($"Average runtime: {averageTime:F2} ms");
    }

    static async Task<double> MeasureParallelOperationsAsync()
    {
        var stopwatch = new Stopwatch();
        stopwatch.Start();

        var task1 = SimulateWorkAsync(200);
        var task2 = SimulateWorkAsync(300);
        var task3 = SimulateWorkAsync(250);
        var task4 = SimulateWorkAsync(150);
        var task5 = SimulateWorkAsync(350);

        await Task.WhenAll(task1, task2, task3, task4, task5);

        stopwatch.Stop();
        return stopwatch.Elapsed.TotalMilliseconds;
    }

    static async Task SimulateWorkAsync(int milliseconds)
    {
        await Task.Delay(milliseconds);

        double result = 0;
        for (int i = 0; i < 10000; i++)
        {
            result += Math.Sqrt(i * Math.PI);
        }
    }
}
```

Kuva 3. Asynkronisten operaatioiden testiohjelma C#-kielellä

```vb
Imports System
Imports System.Diagnostics
Imports System.Threading.Tasks

Module Program

    Sub Main()
        Async().GetAwaiter().GetResult()
    End Sub

    Public Async Function Async() As Task
        Dim iterationCount As Integer = 10
        Dim totalTime As Double = 0

        For i As Integer = 0 To iterationCount - 1
            Dim elapsed = Await MeasureParallelOperationsAsync()
            totalTime += elapsed
            Console.WriteLine($"Lap {i + 1}: {elapsed} ms")
        Next

        Dim averageTime As Double = totalTime / iterationCount
        Console.WriteLine($"Average runtime: {averageTime:F2} ms")
    End Function

    Private Async Function MeasureParallelOperationsAsync() As Task(Of Double)
        Dim stopwatch As New Stopwatch()
        stopwatch.Start()

        Dim task1 = SimulateWorkAsync(200)
        Dim task2 = SimulateWorkAsync(300)
        Dim task3 = SimulateWorkAsync(250)
        Dim task4 = SimulateWorkAsync(150)
        Dim task5 = SimulateWorkAsync(350)

        Await Task.WhenAll(task1, task2, task3, task4, task5)

        stopwatch.Stop()
        Return stopwatch.Elapsed.TotalMilliseconds
    End Function

    Private Async Function SimulateWorkAsync(milliseconds As Integer) As Task
        Await Task.Delay(milliseconds)

        Dim result As Double = 0
        For i As Integer = 0 To 9999
            result += Math.Sqrt(i * Math.PI)
        Next
    End Function
End Module
```

Kuva 4. Asynkronisten operaatioiden testiohjelma VB.NET-kielellä

Taulukko 3. Asynkronisten operaatioiden suoritusajat millisekunteina

| Ajo # | C# | VB.NET |
|---|---|---|
| 1. | 357,18 | 361,09 |
| 2. | 356,88 | 360,37 |
| 3. | 359,10 | 361,71 |
| 4. | 357,90 | 362,33 |
| 5. | 359,26 | 359,73 |
| KA | 358,06 | 361,05 |

Kuten taulukon 3 tuloksista näkyy, oli C# jälleen marginaalisesti nopeampi kieli kuin VB.NET. Tulos ei kuitenkaan tälläkään kertaa ole niin merkittävä, että siitä kannattaisi lähteä vetämään sen isompia johtopäätöksiä.

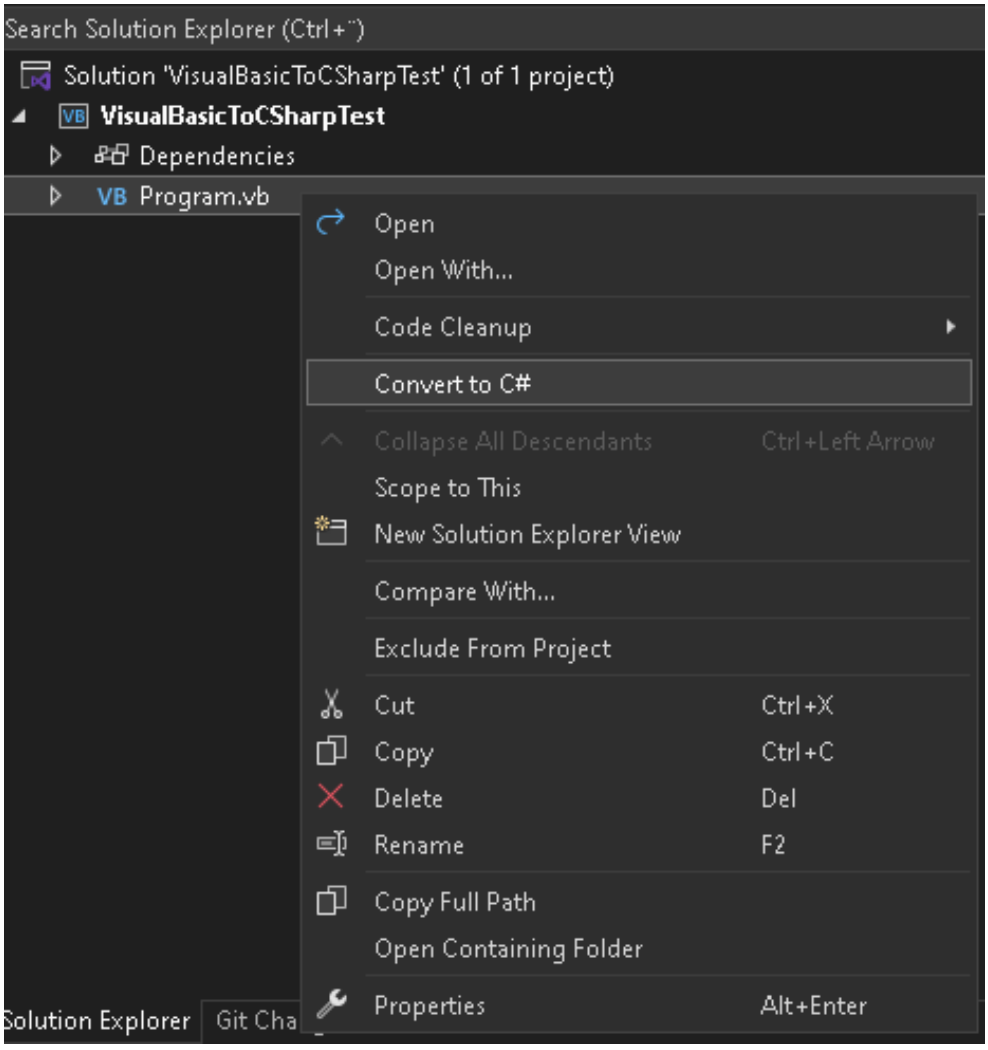
### 4.1.3 Konvertointityökalu CodeConverter

CodeConverter on avoimen lähdekoodin ilmainen ohjelmointikoodin konvertointityökalu. Sen avulla voidaan konvertoida VB.NET-koodia C#-kielelle sekä päinvastoin. FAST on rakentanut oman konvertointityökalun CodeConverterin ympärille, joka ottaa enemmän huomioon muun muassa FAST:n omien kirjastojen konvertointia. FAST:n kehittämä työkalu tekee monia operaatioita koodin konvertoinnin edistämiseen sekä ennen että jälkeen itse konvertoimisen. Tässä luvussa tulen kuitenkin esittelemään CodeConverter-työkalun toiminnallisuutta, koska FAST:n työkalu kuitenkin pohjautuu suurelta osin siihen.

Asensin CodeConverter-työkalun Visual Studio laajennuksena henkilökohtaiselle koneelleni testatakseni sen toimivuutta. Asennus Visual Studio -laajennuksena oli hyvin helppo. Latasin työkalun asennusohjelman Visual Studion kauppapaikalta, jonka jälkeen ajoin asennusohjelman. Ohjelma oli Visual Studion uudelleenkäynnistyksen jälkeen valmis käytettäväksi.

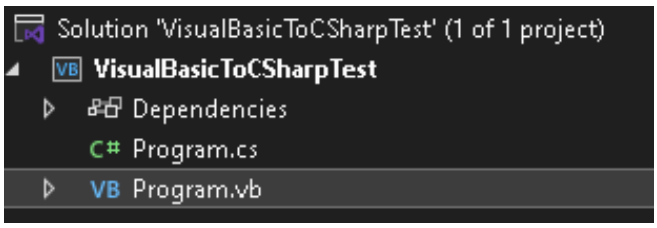
Työkalun käyttö tapahtuu Visual Studion sisällä seuraavasti. Kun olet löytänyt haluamasi VB.NET kielisen luokan, voidaan klikata luokkaa hiiren oikealla

napilla ja etsiä kohdan ”Convert to C#” (kuva 5). Työkalulla voi myös halutessaan kääntää vain pienen osan koodia maalaamalla sen klikkaamalla hiiren oikealla napilla maalattua osaa, ja taas etsimällä kohdan ” Convert to C#”. Myös suuremman kokonaisuuden konvertoiminen kerralla on mahdollista. Työkalulla voi konvertoida kokonaisen Solutionin eli .sln-tiedoston, joka voi sisältää yhden tai useamman projektin. .sln-tiedoston konvertointi toimii samalla tavalla kuin yhden luokan konvertoiminen.




Kuva 5. CodeConverter-työkalun käyttö Visual Studiossa

CodeConverterin ajamisen tuotoksena työkalu luo uuden luokan, jolla on sama nimi kuin alkuperäisellä, mutta se on C#-kielinen (kuva 6).




Kuva 6. Uusi C#-luokka Visual Studion hakemistossa

Testasin CodeConverterin toimivuutta kahdella esimerkillä. Ensimmäisessä esimerkissä (kuva 7) loin hyvin yksinkertaisen laskinluokan VB.NET-kielellä, jossa on muutamia yksinkertaisia operaatioita. Seuraavana on luokan ohjelmakoodi.

```
Public Class Calculator
    Public Function Add(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

    Public Function Subtract(a As Integer, b As Integer) As Integer
        Return a - b
    End Function

    Public Function Multiply(a As Integer, b As Integer) As Integer
        Return a * b
    End Function

    Public Function Divide(a As Integer, b As Integer) As Double
        If b = 0 Then
            Throw New DivideByZeroException("Cannot divide by zero")
        End If
        Return CDbl(a) / b
    End Function

    Public Function CalculateAverage(ParamArray numbers() As Integer) As Double
        If numbers.Length = 0 Then
            Return 0
        End If

        Dim sum As Integer = 0
        For Each num As Integer In numbers
            sum += num
        Next

        Return CDbl(sum) / numbers.Length
    End Function
End Class
```

Kuva 7. Laskinluokka VB.NET-kielellä

CodeConverterin käytön tuloksena sain laskinluokasta (kuva 7) C#-kielisen laskinluokan (kuva 8), jolla on samat toiminnallisuudet kuin alkuperäisellä.

```csharp
using System;

public class Calculator
{
    public int Add(int a, int b)
    {
        return a + b;
    }

    public int Subtract(int a, int b)
    {
        return a - b;
    }

    public int Multiply(int a, int b)
    {
        return a * b;
    }

    public double Divide(int a, int b)
    {
        if (b == 0)
        {
            throw new DivideByZeroException("Cannot divide by zero");
        }
        return a / (double)b;
    }

    public double CalculateAverage(params int[] numbers)
    {
        if (numbers.Length == 0)
        {
            return 0d;
        }

        int sum = 0;
        foreach (int num in numbers)
            sum += num;

        return sum / (double)numbers.Length;
    }
}
```

Kuva 8. CodeConverter-työkalulla käännetty laskinluokka C#-kielellä

CodeConverter suoriutui ensimmäisestä testistä käytännössä täydellisesti, niin kuin sen kuuluikin. Ensimmäisen testin ohjelmakoodi sisälsi hyvin yksinkertaisia operaatioita. Seuraavaksi halusinkin testata CodeConverter-työkalua hieman vaikeammalla esimerkillä. Implisiittiset tyyppimuunnokset ovat mahdollisia VB.NET-koodissa, kun käytetään asetusta ”Option Strict Off”. Implisiittisellä tyyppimuunnoksella tarkoitetaan jonkin tietotyypin automaattista muuttamista

toiseksi ilman ohjelmoijan tekemää erillistä muunnosoperaatiota. Esimerkiksi seuraavanlainen ohjelmakoodi on mahdollinen Option Strict Off -asetuksella.

```
Dim tekstiNumero As String = "123"
Dim numero As Integer = tekstiNumero
Console.WriteLine("Teksti numeroksi: " & numero + 5)  ' Tulostaa 128
```

Kun taas Eksplisiittinen muutos näyttäisi seuraavalta.

```
Dim tekstiNumero As String = "123"
Dim numero As Integer = CInt(tekstiNumero)
Console.WriteLine("Teksti numeroksi: " & numero + 5)  ' Tulostaa 128
```

VB.NET-asetuksella ”Option Strict On” sekä C# vaativat eksplisiittisen muunnoksen. Tästä syystä myös CodeConverter-työkalun kehittäjät suosittelevat Option Strict On -asetuksen käyttöä ennen itse konvertointia (CodeConverter 2024). Halusin kuitenkin testata, miten työkalu pärjäisi implisiittisiä muutoksia sisältävän VB.NET-koodin kanssa. Käänsin alla olevan VB.NET-ohjelman (kuva 9), jossa on erilaisia implisiittisiä muunnoksia.

```vb
Option Strict Off

Module ImplisiittisetMuunnokset
    Sub Main()
        ' Merkkijono numeroksi
        Dim tekstiNumero As String = "123"
        Dim numero As Integer = tekstiNumero
        Console.WriteLine("Teksti numeroksi: " & numero + 5)  ' Tulostaa 128

        ' Numero merkkijonoksi
        Dim luku As Integer = 42
        Dim teksti As String = luku
        Console.WriteLine("Numero tekstiksi: " & teksti & " on merkkijono")

        ' Decimal Doubleksi
        Dim desimaaliluku As Decimal = 123.456D
        Dim liukuluku As Double = desimaaliluku
        Console.WriteLine("Decimal Doubleksi: " & liukuluku)

        ' Double Integeriksi (huomaa tarkkuuden menetys)
        Dim doubleArvo As Double = 45.67
        Dim intArvo As Integer = doubleArvo
        Console.WriteLine("Double Integeriksi: " & intArvo)  ' Tulostaa 46 (pyöristys)

        ' Object-tyypin käyttö
        Dim yleinenMuuttuja As Object = "Teksti"
        yleinenMuuttuja = 42
        Console.WriteLine("Object-tyyppi: " & yleinenMuuttuja + 10)  ' Tulostaa 52

        Console.ReadLine()
    End Sub
End Module
```

Kuva 9. Implisiittisten muutosten testiohjelma VB.NET-kielellä

VB.NET kääntyi C#-kielelle CodeConverter-työkalulla seuraavan kuvan mukaisesti.

```
using System;
using Microsoft.VisualBasic.CompilerServices;

namespace VisualBasicToCSharpTest
{

    static class ImplisiittisetMuunnokset
    {
        public static void Main()
        {
            // Merkkijono numeroksi
            string tekstiNumero = "123";
            int numero = Conversions.ToInteger(tekstiNumero);
            Console.WriteLine("Teksti numeroksi: " + (numero + 5));  // Tulostaa 128

            // Numero merkkijonoksi
            int luku = 42;
            string teksti = luku.ToString();
            Console.WriteLine("Numero tekstiksi: " + teksti + " on merkkijono");

            // Decimal Doubleksi
            decimal desimaaliluku = 123.456m;
            double liukuluku = (double)desimaaliluku;
            Console.WriteLine("Decimal Doubleksi: " + liukuluku);

            // Double Integeriksi (huomaa tarkkuuden menetys)
            double doubleArvo = 45.67d;
            int intArvo = (int)Math.Round(doubleArvo);
            Console.WriteLine("Double Integeriksi: " + intArvo);  // Tulostaa 46 (pyöristys)

            // Object-tyypin käyttö
            object yleinenMuuttuja = "Teksti";
            yleinenMuuttuja = 42;
            Console.WriteLine(Operators.ConcatenateObject("Object-tyyppi: ",
                Operators.AddObject(yleinenMuuttuja, 10)));  // Tulostaa 52

            Console.ReadLine();
        }
    }
}
```

Kuva 10. CodeConverter-työkalulla käännetty implisiittisiä muunnoksia sisältävä ohjelma C#-kielellä

Kuten ohjelmakoodista näkee (kuva 10), CodeConverter on joutunut lisäämään eksplisiittisiä käännöksiä muun muassa Microsoftin Conversions-luokkaa hyväksikäyttäen. CodeConverter on onnistunut saamaan tyyppimuutoksia sisältävän ohjelman toimimaan oikein, mutta koodi on jo hyvin pienessä esimerkissä hieman epäselvempää. Tästä syystä Option Strict On -asetuksen päälle laittaminen sekä sen vaatimien muutosten tekeminen on hyvä idea ennen CodeConverterin käyttöä.

## 4.2 Organisaation näkökulma

Vaikka opinnäytetyöni keskittyy ohjelmointikielen migraatioon tekniseltä kannalta, haluan myös nostaa esiin muutamia organisaation tasolla merkittäviä aiheita migraatioon liittyen. Esimerkiksi ohjelmointikielen ympärillä olevan ekosysteemin laajuus on merkittävä tekijä ohjelmointikielen valinnan kannalta.

### 4.2.1 Kielten ekosysteemit

Kun vertaillaan kahta eri ohjelmointikieltä, on myös tärkeä ottaa huomioon niiden ympärillä olevat ekosysteemit sekä yhteisöt. Mitä isompi yhteisö ja käyttäjämäärä kielellä, sitä varmemmin kehittäjä voi esimerkiksi löytää apua johonkin tiettyyn ongelmaan muilta kehittäjiltä.

Tiobe Software BV:n kehittämän TIOBE-ohjelmointiyhteisöindeksin (Jensen 2025) mukaan vuonna 2024 C# on maailmanlaajuisesti viidenneksi suosituin ohjelmointikieli. Tämän lisäksi C# kruunattiin TIOBE:n vuoden 2023 ohjelmointikieleksi. Palkinto myönnetään kielelle, joka on noussut eniten indeksissä kyseisen vuoden aikana. VB.NET puolestaan on samassa indeksissä sijalla yhdeksän, mikä toisaalta myös todistaa, että se on edelleen hyvin suosittu ohjelmointikieli. Kuitenkin C# on indeksin mukaan yli kaksi kertaa suositumpi kieli kuin VB.NET, mitä voidaan pitää merkittävänä erona kieliä vertaillessa. (Jensen 2025.)

Kielen suosio vaikuttaa moneen tärkeään asiaan. Jo aikaisemmin mainitsemani avun saanti ongelmatilanteessa on helpompaa. Tämän lisäksi mahdollisia työntekijöitä on luonnollisesti enemmän, jos projekti kaipaa syystä tai toisesta lisää osaamista. Suositummalla kielellä on myös helpompi houkutella uusia osaajia.

### 4.2.2 Microsoftin tuki

Microsoft on viime vuosina osoittanut vahvaa sitoutumista C#-ohjelmointikielen kehittämiseen ja pitkäaikaiseen tukeen. Tämä näkyy erityisesti siinä, että kieli saa jatkuvasti uusia ominaisuuksia ja parannuksia, jotka vastaavat modernin

ohjelmistokehityksen tarpeisiin. Esimerkiksi marraskuussa 2023 julkaistu C# 12 toi mukanaan merkittäviä uudistuksia, kuten ensisijaiset konstruktorit ja kokoelmailmaukset, jotka yksinkertaistavat ja tehostavat koodin kirjoittamista ja ylläpitoa. (Microsoft 2024b). Microsoftin aktiivinen panostus C#:n kehittämiseen viestii siitä, että kieli säilyy keskeisenä osana yrityksen teknologiaekosysteemiä myös tulevaisuudessa.

Toisin kuin C#, VB.NET ei ole saanut vastaavaa kehityspanostusta viime vuosina. Microsoftin mukaan VB.NET:n kehitys on siirtynyt ylläpitovaiheeseen, mikä tarkoittaa, että sen tuki rajoittuu lähinnä virheenkorjauksiin ja tietoturvapäivityksiin ilman merkittäviä uusia ominaisuuksia. (Microsoft 2023c). Tämä käytäntö osoittaa, että Microsoft ei enää investoi VB.NET:n pitkäaikaiseen kehitykseen samalla tavalla kuin C#:n. Lisäksi yhtiö ilmoitti elokuussa 2023 lopettavansa Visual Studio for Mac -kehitysympäristön kehittämisen, mikä tarkoittaa, että VB.NET-kehittäjille tarjolla olevat työkalut ja ympäristöt supistuvat entisestään. (Microsoft 2024d.)

Näiden seikkojen perusteella on selvää, että Microsoft priorisoi C#:n kehitystä ja tukea VB.NET:iin verrattuna. C#:n ympärille rakentuu laajempi ja jatkuvasti päivittyvä ekosysteemi, jonka takia se on suositeltavampi valinta sekä nykyisiin että tulevaisuuden ohjelmistokehitystarpeisiin. VB.NET puolestaan näyttää jäävän lähinnä olemassa olevien sovellusten ylläpitoon.

# 5 Tulokset

## 5.1 Teknisten vertailujen tulokset

Opinnäytetyön aikana tehtyjen teknisten vertailujen perusteella voidaan todeta, että C# tarjoaa useita etuja verrattuna VB.NET-kieleen etenkin suurten liiketoimintakriittisten järjestelmien näkökulmasta.

.NET-ohjelmistokehyksen näkökulmasta kielet ovat teknisesti lähes samanarvoisia, sillä molemmat käännetään samanlaiseksi CIL-kieleksi ennen ohjelman ajoa. Kuitenkin C#:n kääntäjän tuottama CIL-koodi on usein hieman

optimoidumpaa, mikä selittää pieniä suorituskykyeroja. Tämä näkyi myös suorituskykytesteissä. Testit osoittivat, että C# suoriutuu laskennallisesti vaativista tehtävistä keskimäärin noin 5 % nopeammin kuin VB.NET. Vaikka ero ei ole valtava, se voi kumuloitua merkittäväksi suurissa järjestelmissä ja erityisesti suorituskykykriittisissä osissa koodia. Mahdollisen suorituskykyerojen merkittävyys migraation toteuttamiselle on loppujen lopuksi kuitenkin pienempi muihin tekijöihin verrattuna.

Syntaksiin liittyvässä vertailussa havaittiin, että vaikka VB.NET tarjoaa verbaalisen ja joidenkin mielestä helpommin lähestyttävän syntaksin, C#:n modernimpi ja tiiviimpi syntaksi on helpommin yhteensopiva muiden nykyaikaisten ohjelmointikielten kanssa. Tämä helpottaa erityisesti nykyaikaisten ohjelmointikäytäntöjen, kuten funktionaalisen ohjelmoinnin ja asynkronisen ohjelmoinnin hyödyntämistä. C#:n case-sensitive-ominaisuus voi myös auttaa välttämään tiettyjä nimeämiseen liittyviä ongelmia, joita VB.NET:in case-insensitive-lähestymistapa voi aiheuttaa.

CodeConverter osoittautui erittäin hyödylliseksi työkaluksi kehittäjän tehdessä ohjelmointikielen migraatiota. CodeConverter kykeni konvertoimaan VB.NET-kielistä koodia ongelmitta C#-kielelle erittäin helposti ja nopeasti. Vaikka opinnäytetyössä tehdyt konversiotestit olivat melko minimaalisia verrattuna migraatioprojektin aikana tehtäviin muutoksiin, CodeConverter-työkalu ja sen ympärille FAST:n itse luoma konversiotyökalu tulee olemaan erittäin suuressa osassa migraatiossa. Tästä huolimatta yksittäisellä kehittäjällä riittää työtä migraatiota tehdessä. VB.NET-koodille pitää tehdä tarvittavat tarkastukset sekä mahdolliset muutokset ennen konversiotyökalun käyttöä. Työkalun käytön jälkeenkin kehittäjän pitää huolellisesti käydä läpi konvertoitu koodi ja sen toiminnallisuudet. Hyvä valmistautuminen muun muassa konversioon liittyvien ohjeistusten ja koulutusten muodossa nousee tärkeään rooliin näin suuressa ja kriittisessä projektissa.

### 5.2 Tulokset organisaation näkökulmasta

Organisaation näkökulmasta siirtyminen VB.NET:stä C#:iin tarjoaa useita pitkän aikavälin etuja. TIOBE-indeksin mukaan C# on yli kaksi kertaa suositumpi kieli kuin VB.NET, mikä tarkoittaa laajempaa yhteisöä, enemmän saatavilla olevia resursseja ja mahdollisesti helpompaa rekrytointia. Suosio näkyy myös erilaisten kirjastojen, työkalujen ja dokumentaation määrässä.

Microsoftin tuen näkökulmasta on selvää, että C# on yhtiön strategisesti tärkeämpi kieli. Microsoft kehittää C#:ia aktiivisesti ja julkaisee säännöllisesti uusia versioita uusilla ominaisuuksilla, kun taas VB.NET on siirtynyt ylläpitovaiheeseen. Tämä tarkoittaa, että C#-pohjaisten järjestelmien tuki ja yhteensopivuus tulevien teknologioiden kanssa on todennäköisesti parempi.

Taloudellisesta näkökulmasta, vaikka migraatio vaatii merkittävää alkuinvestointia työtuntien ja koulutuksen muodossa, pidemmällä aikavälillä se todennäköisesti vähentää ylläpitokustannuksia ja helpottaa järjestelmän jatkokehitystä. C#-pohjaisten järjestelmien laajempi ekosysteemi voi myös vähentää tarvetta räätälöidyille ratkaisuille, mikä voi tuoda kustannussäästöjä pitkällä aikavälillä.

Viimeisimpänä, mutta ei todellakaan vähäisimpänä tekijänä migraatiolla on yksinkertaisesti se, että tuleva GenTax-järjestelmän päivitys on luotu C#-kielelle. Tämä päivitys tulee ennen pitkää olemaan välttämätön uusien ominaisuuksien sekä järjestelmän ylläpidon kannalta.

## 6 Yhteenveto

Tämä opinnäytetyö tarkasteli VB.NET-ohjelmointikielen vaihtamista C#-kieleen suuressa liiketoimintakriittisessä järjestelmässä. Vertailu kattoi sekä teknisiä että organisaationäkökulmia ja pyrki löytämään konkreettisia hyötyjä ja haasteita, joita migraatio aiheuttaa. Työssä myös tarkasteltiin migraation aikana tärkeään rooliin nousevaa koodin konvertointityökalua.

Teknisestä näkökulmasta C# osoittautui suorituskykyisemmäksi vaihtoehdoksi, vaikkakin erot eivät olleet kovin merkittäviä. C#:n modernimpi syntaksi tarjoaa selkeämpiä ja tiiviimpiä rakenteita, jotka helpottavat koodin ylläpitoa ja kehitystä pitkällä aikavälillä. Molempien kielten toimiessa .NET-ohjelmistokehyksen päällä niiden tekninen yhteensopivuus järjestelmän muiden osien kanssa on hyvä.

Opinnäytetyön aikana tehty tutkimus vahvisti myös käsitystä C#:n laajemmasta ekosysteemistä ja yhteisötuesta. Muun muassa TIOBE-indeksin tulokset viime suosilta todistivat C#:n olevan selvästi suositumpi kieli kuin VB.NET. Organisaation näkökulmasta C#:n valinta on selkeästi siis perusteltu. Sen laajempi suosio, jatkuva kehitys ja Microsoftin vahva tuki tekevät siitä turvallisemman valinnan liiketoimintakriittisille järjestelmille. C#-osaajien rekrytointi on myös todennäköisesti helpompaa, ja kielen ympärillä oleva aktiivinen yhteisö tarjoaa laajemmin resursseja kehittäjille.

Migraation haasteet ovat merkittäviä, mutta hallittavissa oikealla suunnittelulla. Suurimmat haasteet liittyvät järjestelmän kokoon, testausvaatimuksiin ja henkilöstön koulutukseen. Näihin haasteisiin kyetään vastaamaan hyvällä valmistautumisella, kehittäjää suuresti auttavalla ohjelmakoodin konvertointityökalulla, monella eri testaustyylillä sekä yhdenaikaisella migraatiolla. Verohallinnossa on jo ennestään käytössä yhden suuremman vuosittaisen järjestelmäpäivityksen malli, joka voisi toimia ohjelmointikielen migraatiota varten erittäin hyvin.

Yhteenvetona voidaan todeta, että vaikka VB.NET-kielestä C#-kieleen siirtyminen on työläs prosessi, sen pitkän aikavälin hyödyt tekevät siitä kannattavan investoinnin. C#:n parempi suorituskyky, modernimpi syntaksi, laajempi yhteisö ja Microsoftin jatkuva tuki takaavat järjestelmän paremman ylläpidettävyyden ja laajennettavuuden tulevaisuudessa. Migraatioprosessin haasteista huolimatta strategisesti suunniteltu siirtymä voi parantaa merkittävästi järjestelmän laatua ja helpottaa sen ylläpitoa pitkällä aikavälillä.

Tulevaisuuden tutkimuskohteina voisi olla hyödyllistä tarkastella erilaisia migraatiostrategioita ja niiden vaikutuksia projektin onnistumiseen, kehittäjien

kokemuksia kielen vaihdoksesta sekä migraation pitkän aikavälin vaikutuksia järjestelmän ylläpidettävyyteen ja jatkokehitykseen.

# Lähteet

CodeConverter. 2024. GitHub-repositorio. Verkkoaineisto. <https://github.com/icsharpcode/CodeConverter>. Luettu 15.4.2025.

Cybellium. 2023. Mastering .NET Framework.

Ecma International. 2001. ECMA-334: C# Language Specification.

Hunter, Scott. 2019. .NET Core is the Future of .NET. Verkkoaineisto. <https://devblogs.microsoft.com/dotnet/net-core-is-the-future-of-net/>. Luettu 12.3.2025.

Lander, Rich. 2019. Introducing .NET 5. Verkkoaineisto. <https://devblogs.microsoft.com/dotnet/introducing-net-5/>. Luettu 22.2.2025.

Microsoft. 2023a. Microsoft .NET language strategy. Verkkoaineisto. <https://learn.microsoft.com/en-us/dotnet/fundamentals/languages>. Luettu 17.12.2024.

Microsoft. 2023b. Overview of .NET Framework. Verkkoaineisto. <https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview>. Luettu 17.12.2024.

Microsoft. 2023c. Elinkaaren UKK – kehittäjätyökalut. Verkkoaineisto. <https://learn.microsoft.com/fi-fi/lifecycle/faq/developer-tools>. Luettu 28.12.2024.

Microsoft. 2024a. Introduction to .NET. Verkkoaineisto. <https://learn.microsoft.com/en-us/dotnet/core/introduction>. Luettu 3.1.2025.

Microsoft. 2024b. The history of C#. Verkkoaineisto. <https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history>. Luettu 19.1.2025.

Microsoft. 2024c. What is Visual Studio? Verkkoaineisto. <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide>. Luettu 6.2.2025.

Microsoft. 2024d. What happened to Visual Studio for Mac? Verkkoaineisto. <https://learn.microsoft.com/en-us/visualstudio/releases/2022/what-happened-to-vs-for-mac>. Luettu 6.2.2025.

Microsoft. 2024e. The .NET Compiler Platform SDK. Verkkoaineisto. <https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/>. Luettu 11.2.2025.

Moghadampour, Ghodrat. 2009. C#-ohjelmointi. Jyväskylä: Docendo.

Mojica, Jose. 2002. C# & VB.NET Conversion Pocket Reference. O'Reilly Media.

Sathiaseelan, J.G.R. & Sasikaladevi, N. 2009. Programming with C# .NET. New Delhi: PHI Learning.

Jansen, Paul. 2025. TIOBE Programming Community index. <https://www.tiobe.com/tiobe-index/>. Luettu 6.3.2025.

Verohallinto. 2019. Verohallinnon digitalisaation kehityshistoria 1947–2019. <https://www.vero.fi/tietoa-verohallinnosta/verohallinnon_esittely/toiminta/kehitt%C3%A4minen/verohallinnon-digihistoria/>. Luettu 28.2.2025.

Vick, Paul. 2004. The Visual Basic .Net Programming Language. Boston: Addison-Wesley. Luettu 9.3.2025.

Wolfram MathWorld. 2025. Münchhausen Number. <https://mathworld.wolfram.com/MuenchhausenNumber.html>. Luettu 8.4.2025.