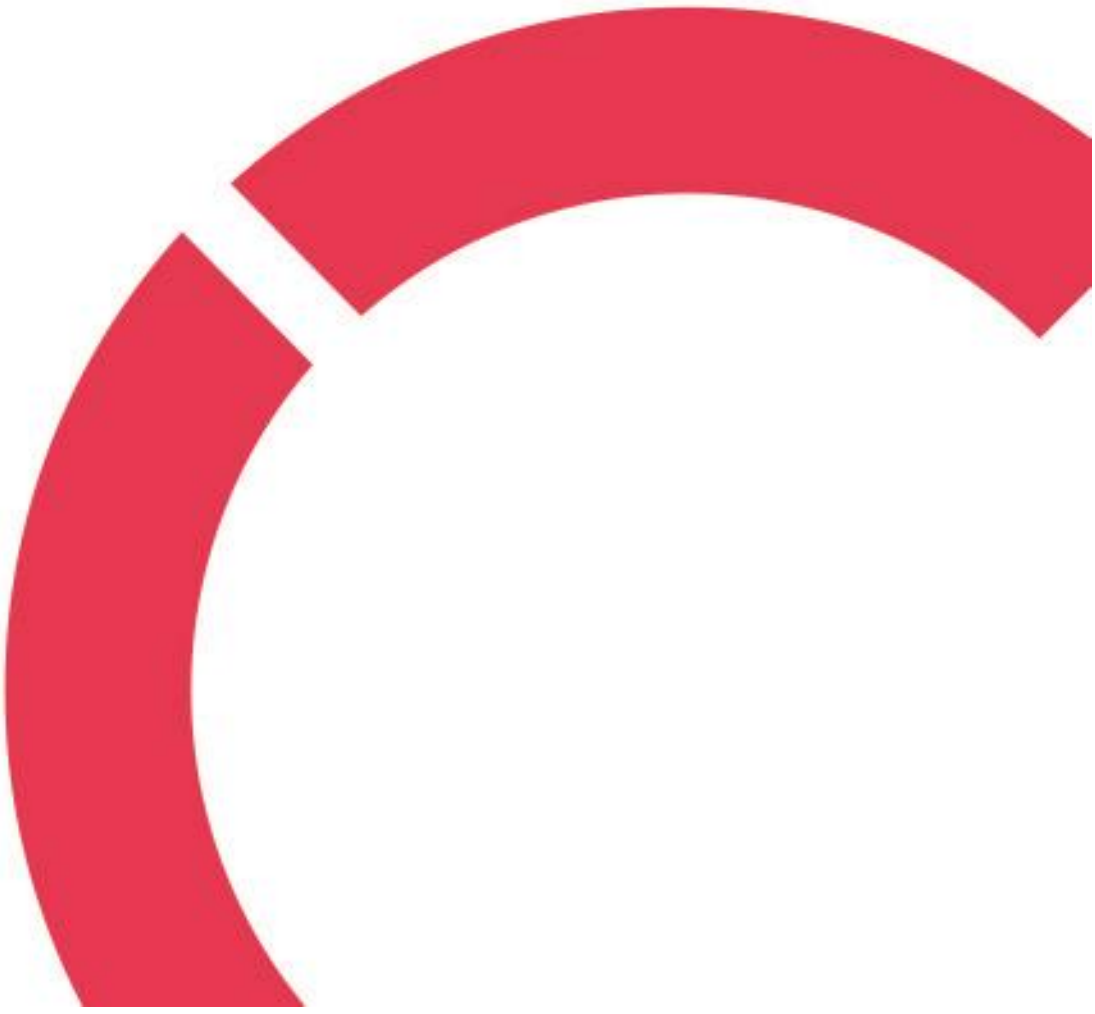
**Alexander Saukkosaari**

# C#-KIELI

**Opinnäytetyö**
**CENTRIA-AMMATTIKORKEAKOULU**
**Tieto- ja viestintätekniikan koulutus**
**Syyskuu 2022**



**TIIVISTELMÄ OPINNÄYTETYÖSTÄ**

| | | |
|---|---|---|
| **Centria-ammattikorkeakoulu** | **Aika**<br>Syyskuu 2022 | **Tekijä/tekijät**<br>Alexander Saukkosaari |
| **Koulutus**<br>Insinööri (AMK), Tieto- ja viestintätekniikka | | ☒ AMK<br>☐ YAMK |
| **Työn nimi**<br>C#-kieli | | |
| **Työn ohjaaja**<br>Jari Isohanni | | **Sivumäärä**<br>34 |
| **Työelämäohjaaja**<br>Eemu Pihlaja | | |

Tämän opinnäytetyön tavoitteena on tutkia C#-ohjelmointikielestä ja sen ominaisuuksista, käydään siitä teoriaa ja käytäntöä läpi. Teoriassa tutkitaan, miten kieli toimii, miten kieltä kehitettiin ja miten se eroaa muista kielistä. Käytännössä tutkitaan, miten kieltä käytetään ohjelmoinnissa. Syvennytään myös siihen, mitkä oikein mahdollistavat kielen toimivuutta, millä kieltä ajetaan ja mitä kielellä voidaan ajaa. Tutkitaan, miten nämä kokonaisuudessaan mahdollistavat ohjelmistotuotannon. Tuloksissa selvitettiin, miten kielellä voidaan tehdä koodia ja ohjelmistoja sekä teorian, että käytännön ymmärrysten kautta. Hyödynnettiin siis saatua tietoa käytäntöön, jolla saatiin ohjelmistotuotantoa aikaiseksi.

**Asiasanat**
Ohjelmistokehys, ohjelmointikieli

**ABSTRACT**

| **Centria University of Applied Sciences** | **Date** September 2022 | **Author** Alexander Saukkosaari |
|---|---|---|
| **Degree programme** Bachelor of Engineering, Information and Communications Technology | | |
| **Name of thesis** C#-language | | |
| **Centria supervisor** Jari Isohanni | | **Pages** 34 |
| **Instructor representing commissioning institution or company** Eemu Pihlaja | | |

The aim of this thesis is to study the C# programming language and its features, we go through theory and practice. In theory, we study how language works, how language was developed and how it differs from other languages. In practice, we study how the language is used in programming. We delve into what enables the functionality of the language, what the language is used to run, and what the language can be used to run. We examine how these enable software production. The results revealed how the language can be used to create code and software through both theory and practical understanding. The knowledge gained was therefore used in practice, which enabled software production to be completed.

**Key words**
Software language, software framework

**TIIVISTELMÄ**
**ABSTRACT**
**SISÄLLYS**

# 1 JOHDANTO

Tämän opinnäytetyön tavoitteena on tutkia Microsoftin C#-ohjelmointikieltä, sen ominaisuuksia ja mahdollisuuksia. Käydään myös sen historiasta, tavoitteista, teoriasta ja käytännöstä asioita läpi. Tutkitaan, miten kieltä on lähdetty kehittämään, miten se toimii, mitä se tarjoaa ja mitä se mahdollistaa. Tutkitaan, miten juuri C# poikkeaa muista ohjelmointikielistä ja miksi juuri sitä kannattaisi käyttää.

C# on yksi C-ohjelmointiperheen kielistä, joihin kuuluu myös itse C-kieli, sekä C++. C# on vähän uudempi kieli näihin verrattuna, sillä se julkaistiin vasta vuonna 1999. C julkaistiin vuonna 1969 ja C++ julkaistiin vuonna 1979. C#-kielen on suunnitellut Anders Hejlsberg ja se ilmestyi ensimmäisen kerran julkiseksi vuonna 2000. C# on tänä päivänä edelleen yksi suosituimmista ohjelmointikielistä. C++-kieleen verrattuna sitä on yksinkertaisempaa käyttää, koska siitä on automatisoitu toimintoja, joita C++-kielellä ei ole. Kielet ovat muuten melko samanlaisia, sillä jos opettelisi yhden näistä kielistä ja samalla perehtyisi ohjelmoinnin perusteisiin, muita kieliä olisi tämän jälkeen helpompaa opetella. (Microsoft 2022c.)

C#-ohjelmointikieltä voidaan käyttää useampiin asioihin; sillä voidaan ohjelmoida mm. sovelluksia, pelejä, mobiilisovelluksia ja verkkosivuja. Microsoft on kehittänyt Visual Studio -nimisen kehitysympäristön, jolla näitä voidaan luoda C#-kielellä. Useampi muu ohjelmointikieli toimii myös Visual Studiossa. Visual Studiolla on myös tukea toimia tietokantojen kanssa, kun tehdään projekteja. Tietokantojen hallintaan kuuluu SQL-niminen kyselykieli, jonka on suunnitellut tietojenkäsittelytieteilijä Donald D. Chamberlin. C#:lla on myös olemassa .NET Framework-niminen ohjelmistokehys. (Microsoft 2022c.)

# 2 .NET FRAMEWORK

C#:iin kuuluu ohjelmistokehys nimeltään .NET Framework, jonka on myös kehittänyt Microsoft. Sitä käytetään Windowsille luoduilla sovelluksilla ja niiden kehitysympäristönä toimii Visual Studio. Ohjelmistokehys itsessään on apuväline ohjelmoinnissa, joka nopeuttaa ohjelmistojen luomista. .NET Framework mahdollistaa useamman ohjelmointikielten yhteensopivuuden toistensa kanssa ja siihen sisältyy komponenttipino, johon kuuluvat FCL (Framework Class Library), LINQ (Language-Integrated Query), ASP.NET (Active Server Pages Network Enabled Technologies), WinForms, sekä CLR (Common Language Runtime). (Microsoft 2022e.)

FCL on luokkakirjasto, johon sisältyy luokkia, tietotyyppejä ja rajapintoja, jotka mahdollistavat tietokantayhteydet, käyttöliittymän- sekä web-kehitykset .NET Frameworkissa. FCL muodostuu kolmesta kategoriasta, jotka ovat kirjoitetut aputoiminnot, käyttöjärjestelmän toiminnanohjaus kääreiden (wrapper) ja rakenteiden (framework) kautta. (Harkiran 2021.)

LINQ:n avulla mahdollistetaan tietojen käsittelyä eri lähteistä C#:ssa, kuten SQL-palvelimesta tai XML:stä. Tällöin näitä lähteitä voidaan käyttää projekteissa. LINQ on API datakyselyille, joissa esiintyy syntakseja SQL:stä. Voidaan kätevästi etsiä tietoa paikoista, joissa toteutetaan IEnumerable-nimistä kokoelmaa, eli rajapintaa. Tämä mahdollistaa rinnastuksen kyselyille. (Dwij 2020.)

Tässä on esimerkki, miten voidaan käyttää LINQ-kyselyä palauttamaan muuttujan tietojen kokoelman:

```
//Tämä on kokoelma lukuja
int[] luvut = new int[] { 4, 5, 6 };

/*Luodaan muuttuja, jonka avulla voidaan hakea kokoelmasta lukuja,
Where-komennon avulla voidaan laittaa ehtoja*/

var tulos = from luku in luvut where luku < 7 orderby luku select
luku;

//Haetaan kaikki luvut muuttujasta
foreach (int i in tulos)
{
    //Syötetään konsolille luvut
    Console.WriteLine(i);
}
```

Tästä tulostuu tällöin konsolille kokoelman luvut 4, 5 ja 6.

ASP.NET on Microsoftin kehittämä palvelinpuolen web-sovelluskehys, jonka avulla voidaan tehdä web-sivustoja ja sovelluksia. Se rakennettiin CLR:lle, minkä ansiosta sitä voidaan käyttää kaikilla .NET-kielillä. Näihin kuuluvat siis C#, F# ja Visual Basic. Sen avulla laajentuvat kirjastot sekä työkalut .NET:ille. Lyhennettynä voidaan siis tehdä web-puolta C#-kielellä. (Microsoft 2022j.)

WinForms on graafinen käyttöliittymä, joka mahdollistaa käyttöliittymän suunnittelun ja kirjoittamisen sovelluksille. Käyttöliittymälle voidaan siis lisätä elementtejä, kuten painikkeita, tekstilaatikoita ja ikkunoita. Näille elementeille voidaan kirjoittaa koodia mikä mahdollistaa niille toimintaa, kun käyttäjä toimii niiden kanssa. Käyttöliittymä siis vastaa käyttäjän tekemiin toimintoihin (Microsoft 2022f).

CLR on sovellusvirtuaalikone, joka mahdollistaa poikkeusten käsittelyä ja muistinhallinnan parantamista, joiden avulla suoritetaan .NET Frameworkin kirjoitettuja ohjelmia. Se kääntää Just-in-Time-ohjelmalla koodin konekäskyksi. Just-in-Time on siis kääntäjä osana CLR:ää ja se suoriutuu CLR:n kanssa tietokoneen prosessorin kautta, joka mahdollistaa .NET Frameworkin toimivuuden. .NET Framework muodostuu FCL:stä ja CLR:stä. (Microsoft 2022e.)

## 2.1 Historia

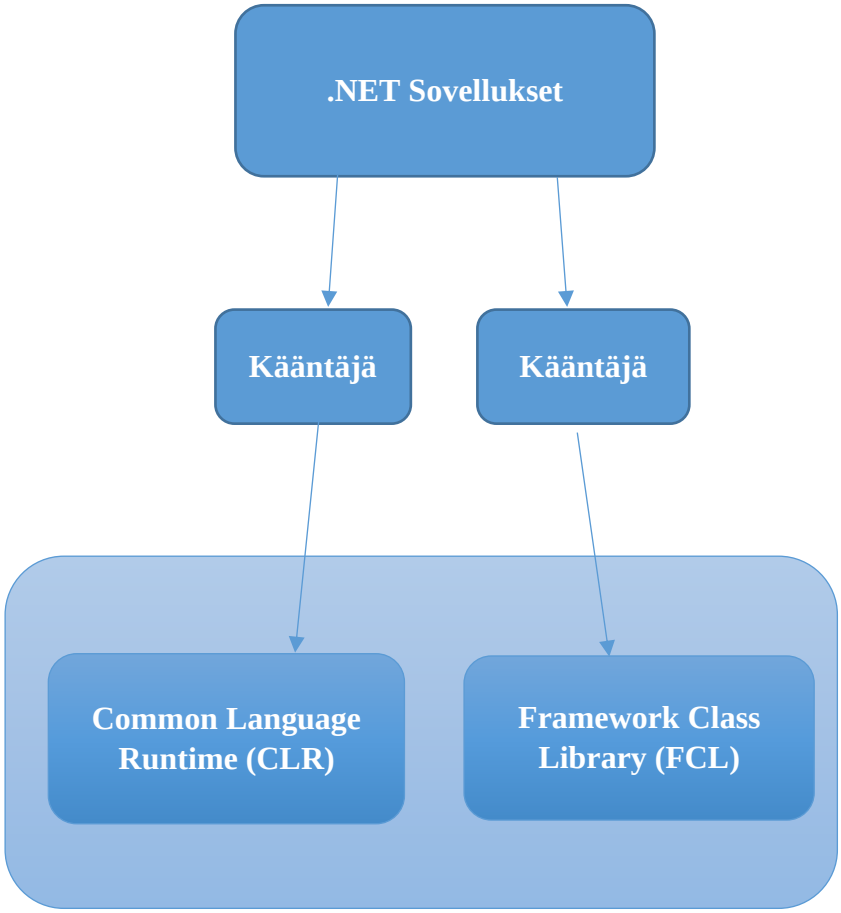
.NET Frameworkin kehittäminen alkoi 1990-luvulla ja silloin sitä kutsuttiin nimellä Next Generation Windows Services. 2000-luvulla julkaistiin ensimmäiset versiot .NET:istä. Ne julkaistiin Windows 98:lle, Windows 2000:lle ja Windows XP:lle. .NET Framework-1.0 version tuki päättyi vuonna 2007. Sen jälkeen julkaistiin .NET Framework 1.1 versio, eli ensimmäinen pieni päivitys. Se julkaistiin vuonna 2003. Muutoksiin kuului mm. .NET Compact Framework, eli pienemmille laitteille tarkoitettu versio .NET Frameworkista. .NET Framework-2.0 versio julkaistiin vuonna 2006 ja tuki päättyi vuonna 2011. Siihen sisältyi useampia muutoksia ja se julkaistiin silloin Visual Studio 2005 kanssa. Tällä hetkellä pyörii .NET Framework-4.8 versio, joka julkaistiin vuonna 2019. Siitä on myös tullut pieni päivitysversio, joka on 4.8.1. Se julkaistiin vuonna 2022 ja se tukee Windows 11 ja Windows 10-käyttöjärjestelmiä. (Microsoft 2022b.)

Vuonna 2002 ECMA hyväksyi C#:n standardiksi kansainvälisesti. ECMA on toimialajärjestö, joka on omistautunut tieto- ja viestintäjärjestelmien standardointiin. ECMA edellytti, että kaikki käyttöönoton kannalta tulisi olla yrityksillä kohtuullisin ja syrjimättömin ehdoin, vaikka Microsoftilla oli paljon yhteistyötä muiden yritysten kanssa. Kyseiset ehdot eivät kuitenkaan viitanneet .NET Frameworkista mm. Windows Formsiin ja ASP.NET:iin. (Microsoft 2022g.)

Vuonna 2007 Microsoft lisensoi .NET Framework-3.5 version lähdekoodit kirjastoista. Tästä tehtiin lähdekoodivarasto, joka julkaistiin vuonna 2008. Se sisälsi mm. ASP.NET:in ja Windows Formsin. Vuonna 2016 Microsoft lisensoi myös Monon uudelleen. Mono on ohjelmistokehys .NET Frameworkille, joka on osa ECMA-standardeja C#:lla. Sen avulla voidaan helposti luoda monialustaisia sovelluksia osana .NET:iä. (Mono Project 2022.)

## 2.2 Käyttökohteet

.NET Frameworkia käytetään monien erilaisten sovellusten, sivustojen ja palveluiden rakentamiseen ja suorittamiseen. .NET:in sovelluksia pystyy suorittamaan useammallakin käyttöjärjestelmillä, eli ei pelkästään Windowsilla. Näistä ovat mm. Linux, macOS, iOS ja Android. .NET Frameworkia sen sijaan käytetään ajamaan .NET-sovelluksia Windowsilla. Kuvassa 1 näytetään, miten .NET sovelluksia suoritetaan. Windowsin asennuksen mukana tulee yleensä .NET Frameworkit mukana, kun näitä tarvitaan tiettyjen sovelluksien ajamiseen. .NET Frameworkia kuitenkin yleensä käyttävät ohjelmistokehittäjät muuhun kuin pelkkien sovellusten ajamiseen. Sitä voidaan siis käyttää lomakepohjaisten sovellusten, verkkopohjaisten sovellusten ja verkkopalvelujen kehittämiseen. .NET-alustalla on saatavilla useita ohjelmointikieliä, joista C# on yleisin. Sitä käytetään sovellusten rakentamiseen Windowsille, puhelimille, webille jne. Se tarjoaa paljon toimintoja ja tukee myös alan standardeja. (Anshul 2022.)




KUVA 1. .NET sovellusten suorittaminen (Harkiran 2021)

## 2.3 Ominaisuudet

.NET Frameworkilla on lukuisia ominaisuuksia, mutta näistä oleellisia ovat mm. FCL (Framework Class Library) ja CLR (Common Language Runtime), jotka yhdessä muodostavat .NET Frameworkin. Siihen sisältyvät myös ASP.NET ja LINQ (Language Integrated Query). FCL ja CLR ovat myös .NET Frameworkin ensimmäisiä toteutuksia (Microsoft 2022a).

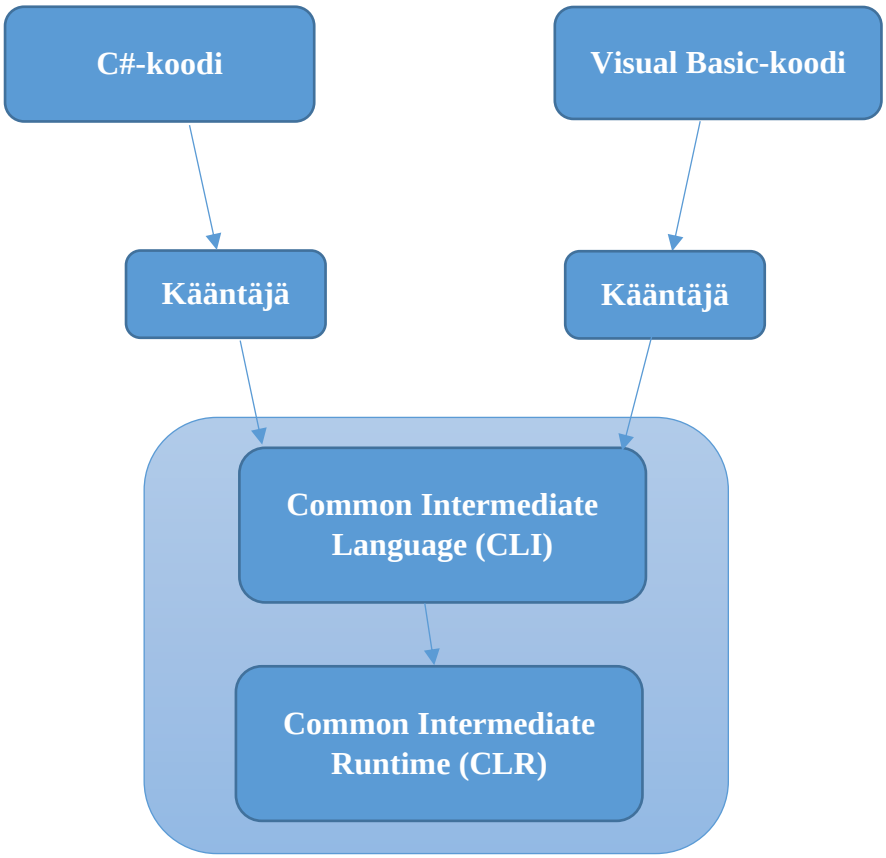
FCL on luokkakirjasto, joka mahdollistaa kielten toimivuuden toistensa kanssa ja CLR on sovellusvirtuaalikone, joka mahdollistaa poikkeusten käsittelyä .NET-ohjelmien suorittamiseen. ASP.NET on verkkosovelluskehys palvelinpuolella, joka mahdollistaa verkkokehityksen tuottamista. LINQ on komponentti, joka mahdollistaa .NET-kieliin kyselyominaisuuksia, eli käsittelee tietokantoja. (Harkiran 2021.)

.NET Frameworkiin kuuluu myös kokoonpano CLI (Common Language Infrastructure), joka mahdollistaa käyttöönoton ja versioinnin. Näihin kuuluvat EXE ja DLL. EXE, jotka mahdollistavat prosesseja, joissa käytetään DLL-luokkia. Kokoonpano saattaa koostua useammastakin tiedostosta ja näistä olevia kooditiedostoja nimetään moduuleiksi. Koodimoduuleita voi siis myös koostua useammassakin kokoonpanossa. CLI-kokoonpanoihin ilmaantuu koodia, joka on peräisin CIL:stä (Common Intermediate Language), jota tuodaan CLI:stä ja käytetään JIT (just-in-time) sen kääntämiseen konekieleksi. Tämä toimii CLR:n (Common Language Runtime) kautta .NET Frameworkilla. (Microsoft 2022a.)

## 2.4 .NET

C#:iin kuuluu myös toinen Microsoftin kehittämä ohjelmistokehys nimeltään .NET (aiemmin .NET Core), joka on seuraaja .NET Frameworkille. Se toimii Windowsilla, Linuxilla, sekä macOS-käyttöjärjestelmillä .NET Core 1.0 julkaistiin ensimmäisen kerran vuonna 2016, joten se on vielä melko uusi. Tällä hetkellä on menossa .NET 6, joka julkaistiin vuonna 2021. .NET tukee mm. C#-kieltä ja Visual Basicia. .NET käyttää myös CLI:tä ja CLR:ää. (Setia 2020.)

.NET tukee myös ASP.NET-sovelluksia, kirjastoja ja Windows Formsia. Kuvassa 2 näytetään, miten koodeja suoritetaan CLI:llä ja CLR:llä. Työpöytäteknologiat (WinForms ja WPF) eivät alun perin toimineet. NETillä, mutta .NET Core 3.0-versio mahdollisti näiden kirjoituksen .NETillä. .NET tukee myös NuGet-paketin käyttöä, jota .NET Framework taas ei tue. NuGet-paketti mahdollistaa koodin jakamisen eteenpäin, joten sitä voidaan uudelleen käyttää. .NETin komponentteihin kuuluvat CoreCLR ja CoreFX. CoreCLR on .NET Coren ajoaika, joka sisältää useita matalan tason luokkia. CoreFX taas on tehty perusluokan kirjastoista. (Setia 2020.)




KUVA 2. Koodien suoritus CLI:llä ja CLR:llä (Jain 2017)

## 2.5 .NET 6.0

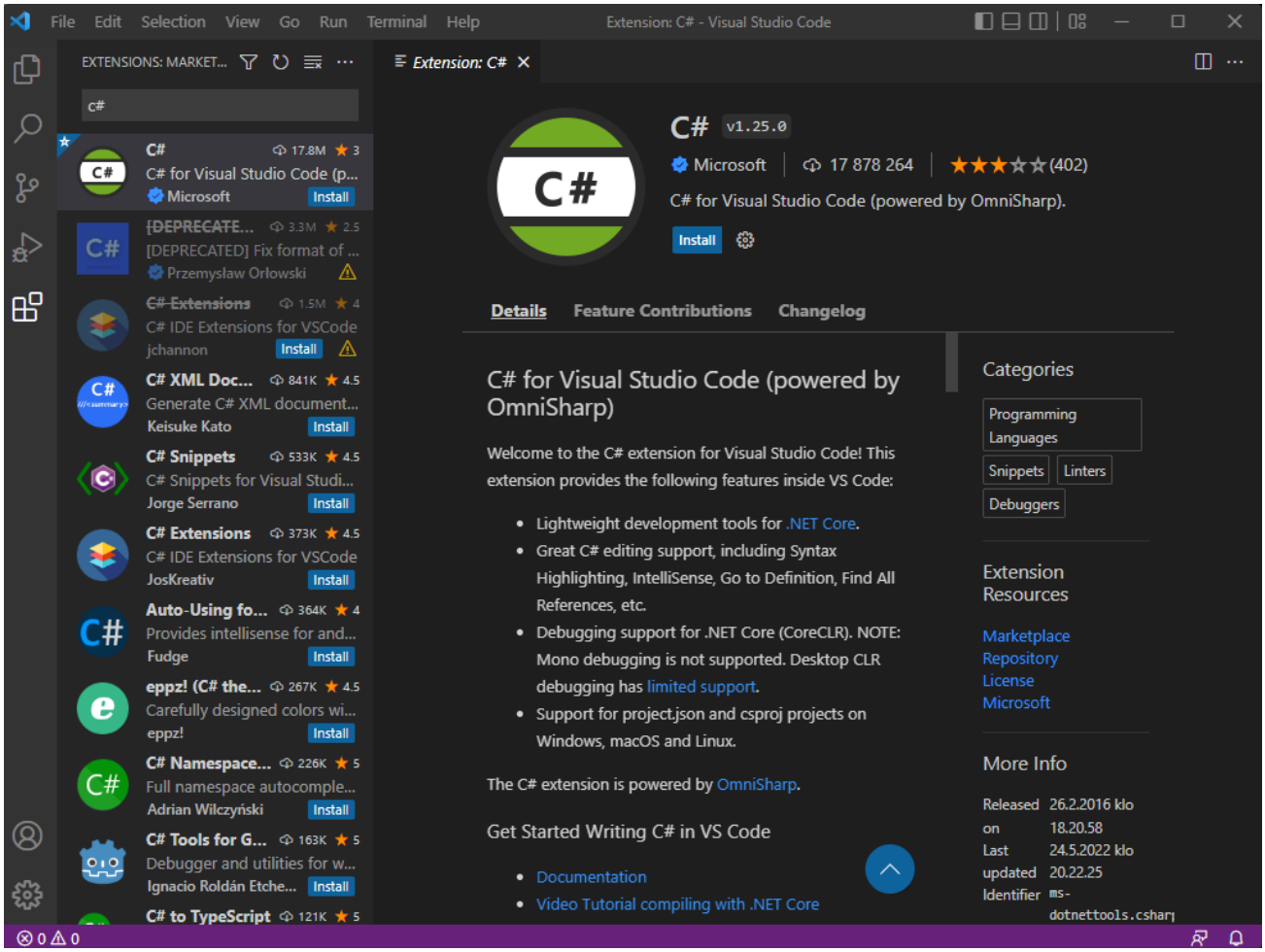
.NET 6.0 on .NETin uusin versio tällä hetkellä, joka julkaistiin vuonna 2021. Sen tuomiin muutoksiin kuuluu mm. yksinkertaistettu kehitys, joka mahdollistaa käyttäjälle helpomman aloituksen tämän version parissa. Suorituskykyä on myös parannettu, minkä ansiosta .NET 6 on juuri nyt nopein verkkokehys. .NET 6 tarjoaa mm. uusia git-työkaluja, kehittyneen koodialustan ja testaustyökaluja. Git on siis versiohallintaan perustuva ohjelmisto, jonka kautta voidaan kätevästi koordinoida työtä. Sitä tuetaan kuitenkin vain kolmen vuoden ajan viimeisimpänä julkaisuna, eli tuen päädyttyä julkaistaan uusi .NET versio, joka tällöin on .NET 7. Tällä hetkellä Visual Studio 2022 tukee .NET 6:ta. (Chand 2022.)

.NET 6 toi mukanaan paljon uusia muutoksia, joista mm. FileStream, PGO (Profile Guided Optimization ja AOT (ahead-of-time) saivat muutoksia. System.IO.FileStreamia on koodattu uudelleen suorituskyvyn parantamiseksi .NETissä ja sitä voidaan nyt luoda Windowsille I/O:lle. .NET 6 tarjoaa myös uudehkon PGO:n, nimeltään Dynaaminen PGO. Se optimoi koodia entistä nopeammin aiempaan verrattuna. Siihen sisältyy myös Crossgen 2 työkalu, mikä mahdollistaa AOT:in parantamisen, eli nopeuttaa sovelluksen käynnistämistä. Tämä uusi Crossgen 2 työkalu kirjoitettiin C#-kielellä, toisinkuin erilleen työkalun ensimmäinen versio, joka kirjoitettiin sen sijaan C++-kielellä. (Microsoft 2022k.)

Lisäksi .NET 6.0:aan sisältyy Hot Reload-niminen ominaisuus, joka mahdollistaa pidettäessä ohjelmaa auki ja muokattaessa koodia sen, että painamalla Hot Reload nappia, jonka kautta uudet koodimuutokset menevät ohjelmaan mukaan ilman, että käyttäjän tarvitsee sammuttaa ja aloittaa ohjelmaa uudestaan uusimmilla koodimuutoksilla, joka huomattavasti säästää aikaa. (Microsoft 2022k.)

## 2.6 .NET 6.0 käyttöönotto

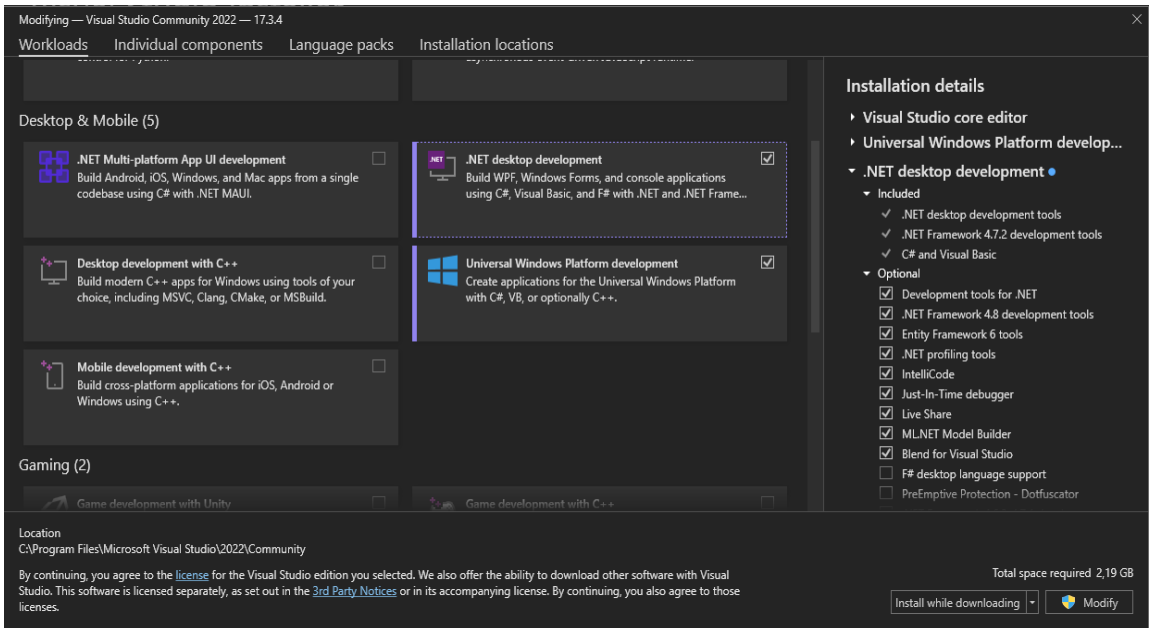
.NET 6.0 voidaan ladata osoitteesta https://dotnet.microsoft.com/en-us/download?WT.mc_id=hello-world-17228-cxa. .NET 6.0 tarvitsee joko Visual Studion tai Visual Studio Coden (C#-puoli) myös asennetuksi. Visual Studio 2022 voidaan ladata osoitteesta https://visualstudio.microsoft.com/vs/. Visual Studio Code voidaan ladata osoitteesta https://code.visualstudio.com/ ja asennuksen jälkeen voidaan ladata C#-laajennus. Visual Studio Codessa painetaan näkymässä Extensions ikkuna auki ja sen kautta voidaan asentaa Microsoftin C#-laajennus. Tämä esitetään kuvassa 3.




KUVA 3. Visual Studio Code C#-laajennus (Microsoft 2022i)

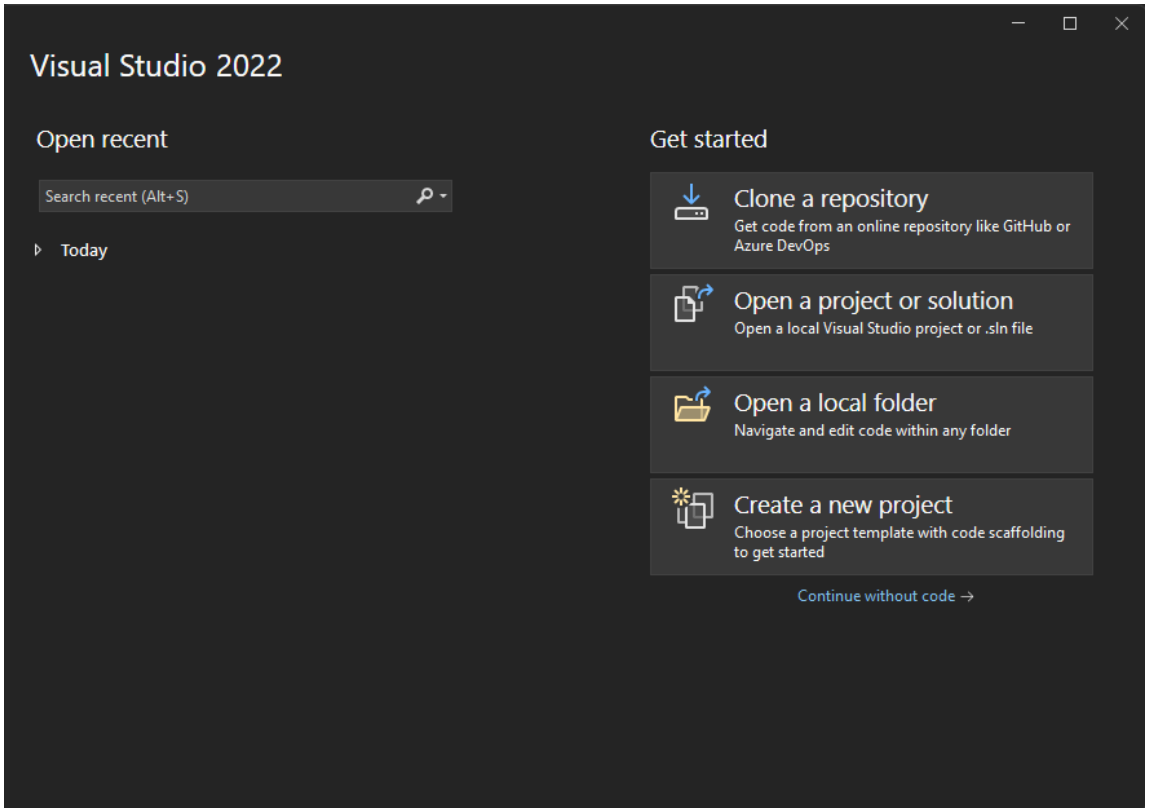
Kun .NET 6.0 on asennettu, se siirtyy valmiiksi Visual Studioon ja pitää valita käyttöön, mitä kuvaan myöhemmin. Peruste sille, miksi .NET 6.0:aa kannattaa hyödyntää on se, että .NET 6.0 on aiempiin versioihin verrattuna paljon nopeampi ja sisältää enemmän ominaisuuksia ja tukea. Visual Studion

asennuksen aikana tulee myös näkymä, josta voidaan valita useampi vaihtoehto laajennusten asentamisiin, näistä esimerkkejä ovat .NET desktop development ja Universal Windows Platform development-paketit. Tämä esitetään kuvassa 4.



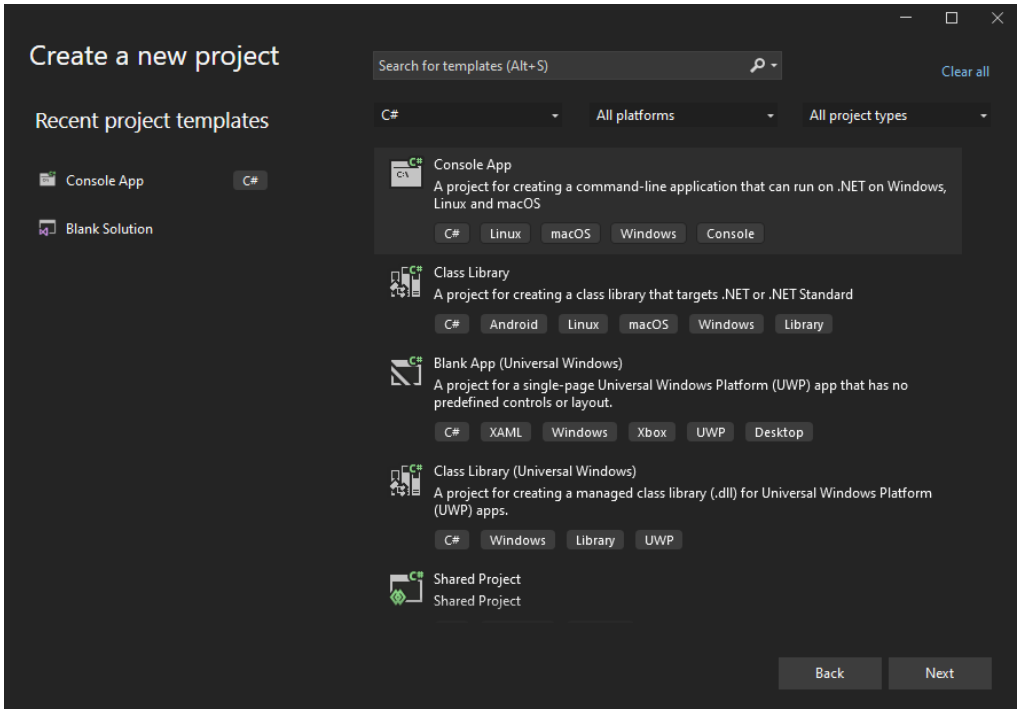

KUVA 4. Visual Studio asennuspaketit (Microsoft 2022h)

Kun luodaan omaa projektia, voidaan valita, ladataanko pilvestä jonkun projektin tiedot vai aloitetaanko tyhjästä uutta. Näille on neljä vaihtoehtoa, kun avataan Visual Studion näkymä. Tämä esitetään kuvassa 5.



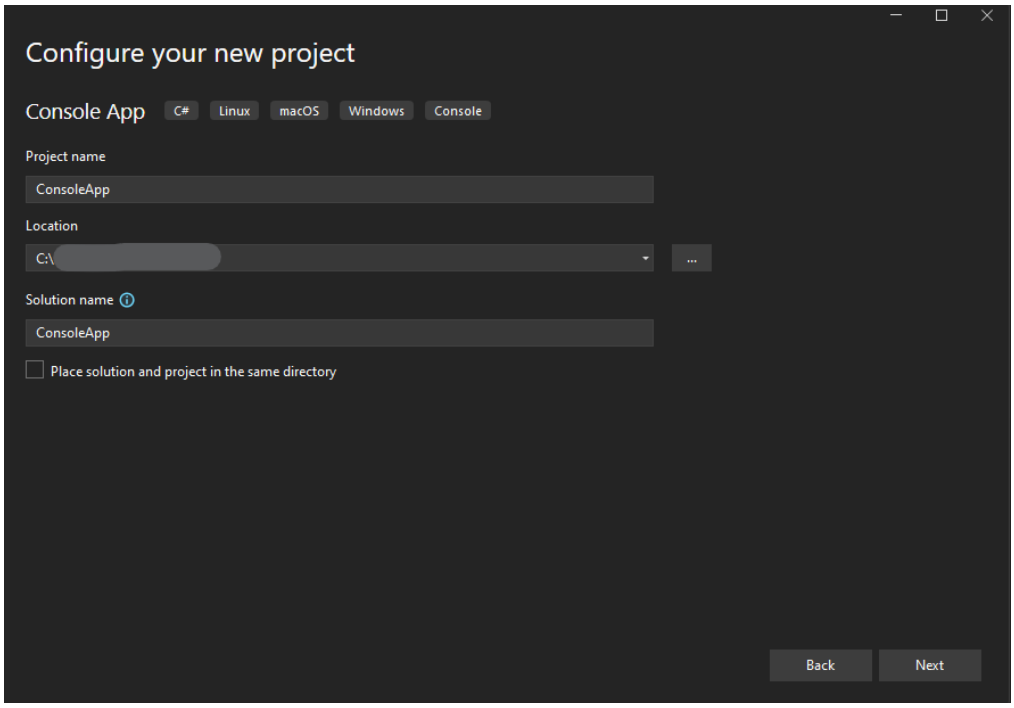
KUVA 5. Visual Studio aloitusnäkymä (Microsoft 2022h)

Jos luodaan esimerkkinä uutta tyhjää projektia, painetaan ”Create new project”, josta siirrytään näkymään, josta voidaan valita mm. haluttu C#-projekti. Console App on oikein oiva vaihtoehto, jos halutaan harjoitella ohjelmointia. Tämä näkymä esitetään kuvassa 6.



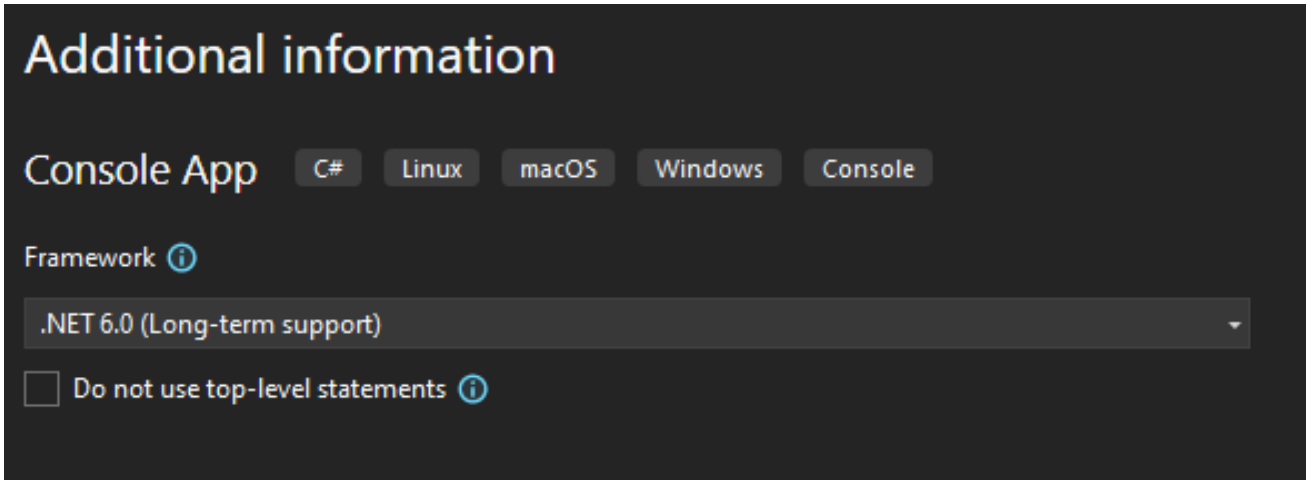

KUVA 6. Visual Studio projektin luonti (Microsoft 2022h)

Kun valitaan projekti, niin sille voidaan antaa oma nimi ja paikka, jonne tiedosto tallennetaan. Tämä esitetään kuvassa 7.



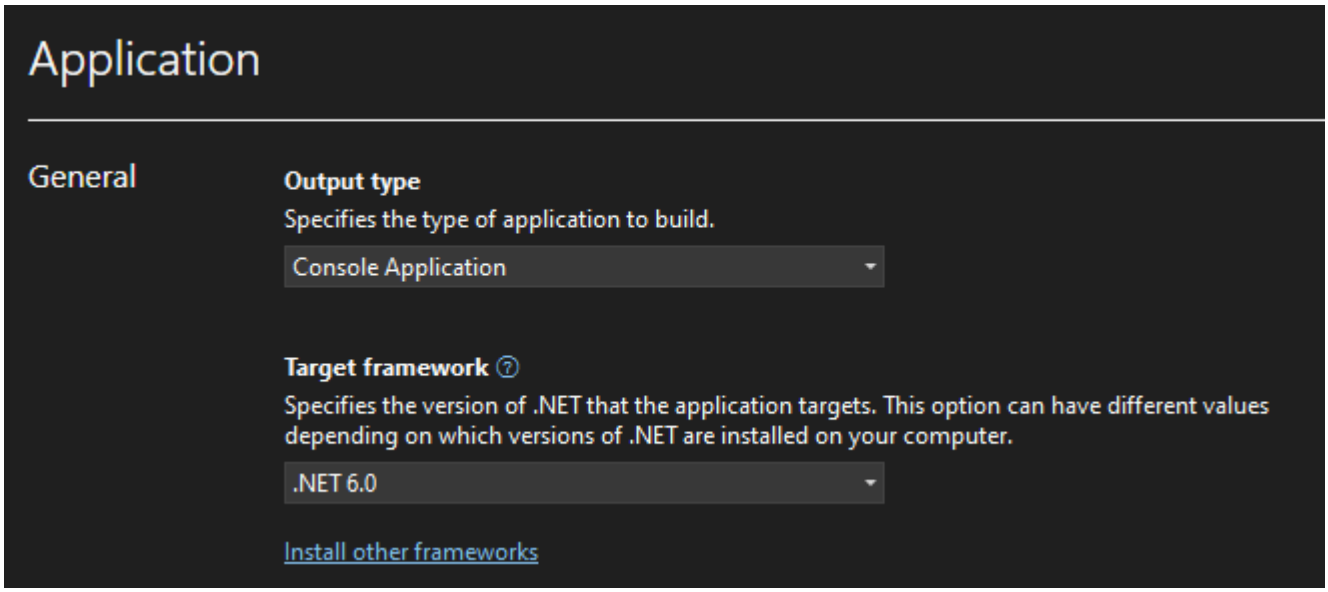

KUVA 7. Visual Studio projektin laatiminen (Microsoft 2022h)

Kun kyseiset askeleet on tehty, tulee ”Additional information”-ikkuna, **josta voidaan valita .NET 6.0 käyttöön omaan projektiin.** Jotta ohjelma toimii tätä varten, tulee .NET desktop development-laajennus olla asennettuna, mikä näkyy ylhäällä olevassa kuvassa 7. Tällöin voidaan myös rakentaa omaa projektia .NET 6:tta hyödyntäen. Tämä esitetään kuvassa 8.




KUVA 8. Visual Studio kehyksen valitseminen (Microsoft 2022h)

Valmiissa projekteissa voidaan vaihtaa .NET 6.0-versioon joko koodin kautta tai projektin asetusten kautta. Kun painetaan hiiren oikealla projektia, valitaan joukosta properties, niin ilmestyy ”Applications” näkymä, josta voidaan Target framework valikon kautta valita haluttu ohjelmistokehys projektille. Tämä esitetään kuvassa 9.




KUVA 9. Visual Studio kehyksen vaihtaminen (Microsoft 2022h)

# 3 OHJELMOINTI

C# (C-Sharp) on Microsoftin kehittämä ohjelmointikieli, joka toimii .NET Frameworkilla. C# käytetään mm. web-kehitykseen, ohjelmistokehitykseen ja pelikehitykseen. C# on yksi jäsenistä C-kieliperheestä, joissa on myös itse C ja C++. Kielet ovat siis ideoiltaan samoja, mutta niissä on omat eronsa. C#:lla tietyt toiminnot ovat automaattisia, mitä esimerkiksi C++-kielellä eivät ole. C# käytetään myös enemmän ohjelmistokehitykseen, toisin kuin C++ käytetään enemmän konsolisovelluksissa (Microsoft 2022d).

**Hello World C#:lla:**

```
namespace HelloWorld
{
    class Hello
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

**Hello World C++:lla:**

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World!\n";
}
```

Console.WriteLine, sekä cout << tulostavat syötettyjä arvoja. Vaikka C++ -koodi näyttää tässä tilanteessa lyhyemmältä, niin isommissa projekteissa se ei välttämättä ole lyhyempää. C#-koodi on tuossa tilanteessa pidempi, koska se käyttää luokkia. C# itse on siis olio-ohjelmointikieli. Olio-ohjelmointi koostuu siis objekteista, luokista, metodeista, sekä attribuuteista. Luokilla voidaan luoda objekteja (Microsoft 2022d).

**Luokka objektilla C#:lla:**

```
namespace HelloWorld
{
    //Tämä on luokka
    class Earth
    {
        //Tämä on objekti
        string country = "Finland";
    }
}
```

Ohjelmoinnissa on myös tärkeää laittaa koodiin kommenttia // tai /* */ komennoilla, jotta ymmärtää koodia paremmin. //-komento pyyhkii yhden rivin, mutta /* */-komentoa voi käyttää useampaan riviin.

### 3.1 Kutsumiset

Attribuuteilla tarkoitetaan tietojen yhdistämistä koodiin, siis menetelmiä, tyyppejä ja kokoonpanoja. Metodit ovat koodilohkoja, jotka sisältävät niiden sisälle kirjoitettua koodia ja niitä voidaan myöhemmin kutsua ohjelmassa uudelleen. Niitä kutsutaan myös funktioiksi ja sisältävät sulut. (Microsoft 2022d.)

**Attribuutti metodilla C#:lla:**

```
namespace HelloWorld
{
    //Tämä on attribuutti
    [Serializable]
    class Earth
    {
        //Tämä on metodi
        public void Countries()
        {
            string country = "Finland";
        }
    }
}
```

Kun metodeja kutsutaan myöhemmin koodissa, voidaan niiden sisältöjä kutsua uudelleen, ettei tarvitse samaa koodia kirjoittaa uudelleen useampaan kertaan. Luokista pystytään myös kutsumaan muuttujia.

**Metodin kutsuminen C#:lla:**

```
namespace HelloWorld
{
    class Earth
    {
        /*Annetaan funktion objektille arvo,
        jota voidaan kutsua muualla*/
        public static string Countries()
        {
            string country = "Finland";
            return country;
        }

        /*Kutsutaan pääfunktiossa maata, niin
        tulostuu palautettu arvo*/
        public static void Main(string[] args)
        {
            Console.WriteLine(Countries());
        }
    }
}
```

**Tulostuu:**

Finland

Myös luokkia voidaan kutsua myöhemmin koodissa C#:lla sieltä sisältäviä metodeja ja objekteja. On siis monta tapaa päästä käsiksi aiempiin tehtyihin koodeihin ilman, että tarvitse useampaan kertaan kirjoittaa samaa asiaa.

**Luokan kutsuminen C#:lla:**

```
namespace HelloWorld
{
    class Earth
    {
        /*Annetaan funktion objektille arvo,
        jota voidaan kutsua muualla*/
        public string Countries()
        {
            string country = "Finland";
            return country;
        }
    }

    class Universum
    {
        public static void Main(string[] args)
```

```
        {
            //Kutsutaan earth-luokkaa
            Earth earth = new Earth();

            //Tulostetaan earth-luokan sisältö
            Console.WriteLine(earth.Countries());
        }
    }
}
```

**Tulostuu:**

Finland

### 3.2 Ehdot

Ohjelmoinnissa voidaan myös kätevästi asettaa ehtoja, jos halutaan tiettyjä toimintoja suoriutuvan ehtojen mukaisesti. Voidaan hyödyntää aiempaa koodia tämän toteuttamisessa.

**Ehdot C#:lla:**

```
namespace HelloWorld
{
    class Countries
    {
        public static void Main(string[] args)
        {
            //Kysytään ohjelmassa käyttäjältä vastaamaan
            Console.WriteLine("Where does santa live?");

            //Tämä lukee mitä käyttäjä on syöttänyt konsoliin
            string answer = Console.ReadLine();

            /*Jos-komento tarkistaa, onko vastaus "Finland",
            joka tulostuu tuosta funktiosta*/
            if (answer == "Finland")
            {
                Console.WriteLine("Yes, santa lives in " + answer +
"."); 
            }

            /*Muulloin-komento tarkistaa, onko
            vastaus jotain muuta kuin "Finland"*/
            else
            {
                Console.WriteLine("No, santa does not live in " +
answer + ".");
            }
        }
    }
}
```

**Tulostuu:**

Where does santa live?

Finland

Yes, santa lives in Finland

**Tai:**

Where does santa live?

Norway

No, santa does not live in Norway.

### 3.3 Silmukat

Ohjelmoinnissa voidaan myös käyttää silmukoita koodin toistamisiin loputtomasti tai päättämällä halutessaan. Voidaan taas hyödyntää aiempaa tehtyä koodia tätä varten.

**Silmukat C#:lla:**

```
namespace HelloWorld
{
    class Countries
    {
        public static void Main(string[] args)
        {
            //Kysytään ohjelmassa käyttäjältä vastaamaan
            Console.WriteLine("Where does santa live?");

            /*Silmukka, joka pyörittää koodia niin kauan,
            kunnes "break" pysäyttää koodin*/
            while (true)
            {
                //Tämä lukee mitä käyttäjä on syöttänyt konsoliin
                string answer = Console.ReadLine();

                /*Jos-komento tarkistaa, onko vastaus "Finland",
                joka tulostuu tuosta funktiosta*/
                if (answer == "Finland")
                {
                    Console.WriteLine("Yes, santa lives in " + an-
swer + ".");
                    break;
                }

                /*Muulloin-komento tarkistaa, onko
                vastaus jotain muuta kuin "Finland"*/
                else
                {
                    Console.WriteLine("No, santa does not live in "
+ answer +
                    ". Guess again.");
                }
            }
        }
    }
}
```

**Tulostuu:**

Where does santa live?

Norway

No, santa does not live in Norway. Guess again.

Finland

Yes, santa lives in Finland.

### 3.4 Taulukot

Ohjelmoinnissa voidaan käyttää taulukoita säilyttämään sisällä olevia objekteja, joita voidaan myös halutessaan kutsua ohjelmassa myöhemmin.

**Taulukot C#:lla:**

```
namespace HelloWorld
{
    class Countries
    {
        public static void Main(string[] args)
        {
            //Tämä on kokoelma taulukon sisällöstä
            string[] countries = { "Finland", "Norway", "Sweden",
"Denmark" };

            /*Tällä voidaan hakea kaikki sisällöt taulukosta
            ja halutessaan tulostaa kaikki kerralla*/
            foreach (var items in countries)
            {
                Console.WriteLine(items);
            }

            /*Tällä voidaan hakea erikseen taulukosta
            tietyn elementin, tässä otetaan ensimmäinen elementti*/
            Console.WriteLine("Haetaan erikseen: " + countries[0]);
        }
    }
}
```

**Tulostuu:**

Finland

Norway

Sweden

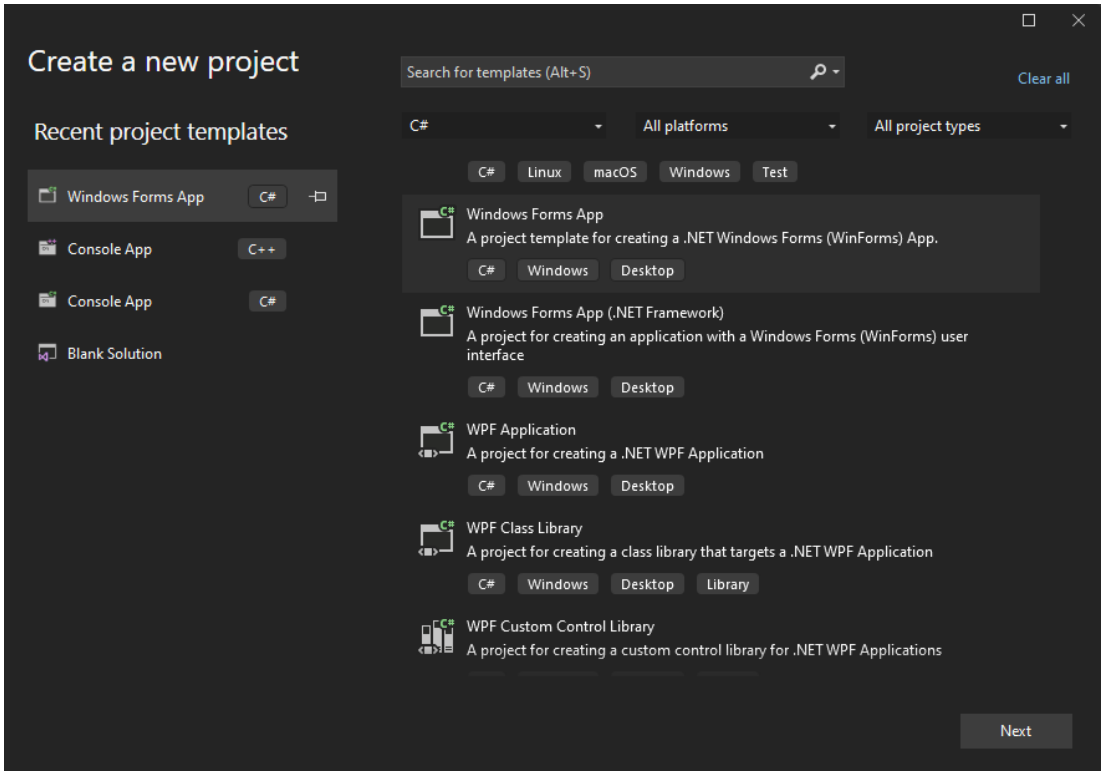
Denmark

**Tai:**

Haetaan erikseen: Finland

# 4 KÄYTTÖLIITTYMÄT

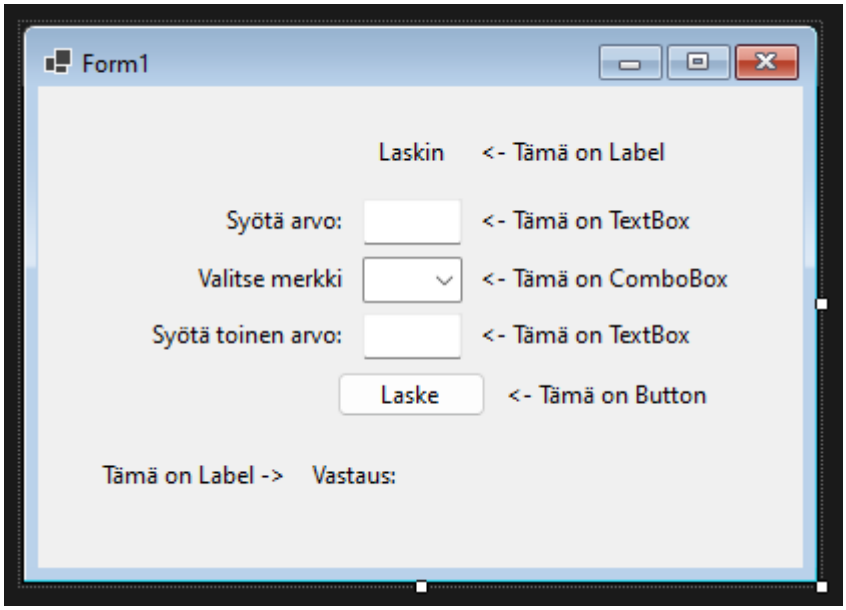
C#-kieltä voidaan käyttää käyttöliittymien rakentamiseen koodin kautta, sekä Visual Studion kautta voidaan rakentaa käyttöliittymiä Windows Formsilla. Käyttöliittymään voidaan lisätä vaikka mitä, kuten nappuloita, tekstilaatikoita, otsikoita ja muuta. Koodissa voidaan tehdä näille toimintaa, kun käyttäjä toimii käyttöliittymän kanssa. Jotta päästään tätä toteuttamaan, luodaan uusi projekti ja valitaan Windows Forms App. Tämä esitetään kuvassa 10.




KUVA 10. Visual Studio projektin luonti (Microsoft 2022h)

Kun on valittu projekti ja annettu sille nimi, voidaan lähteä rakentamaan käyttöliittymää. Näkymässä vasemman yläkulman lähellä on ToolBox, josta voidaan valita käyttöliittymälle lisättäviä tavaroita. Voidaan esimerkiksi lähteä tekemään laskinta; ensimmäiseksi suunnitellaan sitä käyttöliittymään, lisätään otsikko ja tekstit, lisätään nappulat ja tekstikentät. Tämän jälkeen lähdetään tekemään tälle koodia, kun toimitaan näiden kanssa.

Tämä esitetään kuvassa 11.




KUVA 11. Visual Studio Windows Forms esikatselu (Microsoft 2022h)

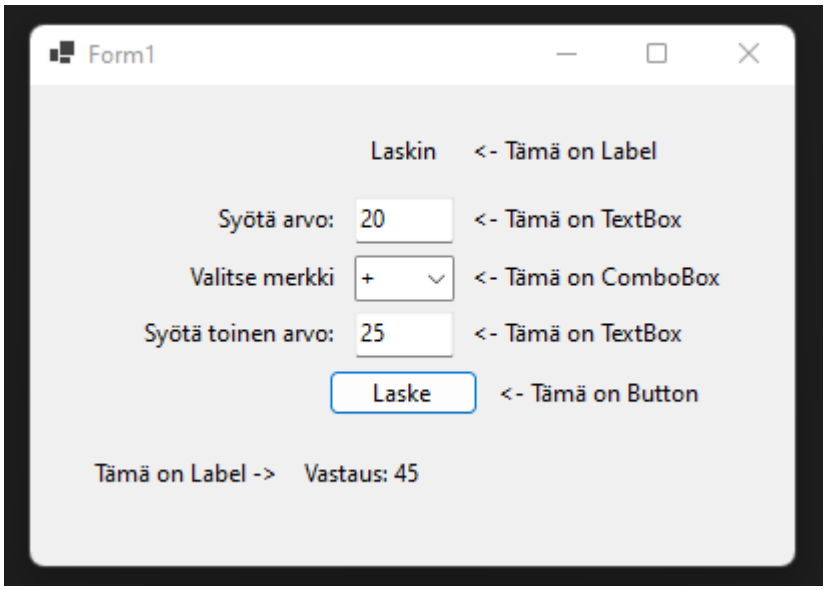
Koodia voidaan kätevästi lisätä toimimaan tuon napin kanssa, eli kun käyttäjä painaa nappia, niin sen kautta voidaan laittaa arvot laskemaan syötettyjen arvojen perusteella. Kun syötetään järjestyksessä ensimmäiseen tekstilaatikkoon luku, valitaan merkki, jolla laskentaan luvut, syötetään toiseen tekstilaatikkoon luku ja painetaan laske-nappia, niin ”Vastaus”-tekstin viereen tulostuu laskettu arvo. Tätä varten voidaan käyttää seuraavaa koodia:

```
namespace WindowsForms
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        //Tämä funktio suorittaa koodin kun painetaan nappia
        private void countButton_Click(object sender, EventArgs e)
        {
            /*Haetaan ensimmäisestä syötetystä
            textboxista luku*/
            int value = int.Parse(textBox1.Text);
            /*Haetaan toisesta syötetystä
```

```
            textboxista luku*/
            int value2 = int.Parse(textBox2.Text);
            /*Jos valitaan ensimmäinen merkki
            comboboxista, sitä käytetään laskuun*/
            if (selectBox.SelectedIndex == 0)
            {
                //Vastaus-labelin viereen tulostuu syötettyjen arvo-
jen summa
                answerLabel.Text = "Vastaus: " + Con-
vert.ToString(value + value2);
            }
        }
    }
}
```

Koodissa on tällä hetkellä vain yhteenlaskulle ehto, koska muilla ehdoilla koodista tulee melko pitkä, eli käyttöliittymässä näkyy vain yhteenlaskun tulos. Tämä esitetään kuvassa 12.




KUVA 12. Visual Studio Windows Forms-sovellus (Microsoft 2022h)

Lisätään samaan koodiin ehdot miinus-, kerto-, sekä jakolaskuille. Ehdot näyttävät tältä:

```
/*Jos valitaan toinen merkki
comboboxista, sitä käytetään laskuun*/
else if (selectBox.SelectedIndex == 1)
{
    //Vastaus-labelin viereen tulostuu syötettyjen arvojen erotus
    answerLabel.Text = "Vastaus: " + Convert.ToString(value -
value2);
}

/*Jos valitaan kolmas merkki
```

```
comboboxista, sitä käytetään laskuun*/
else if (selectBox.SelectedIndex == 2)
{
    //Vastaus-labelin viereen tulostuu syötettyjen arvojen tulo
    answerLabel.Text = "Vastaus: " + Convert.ToString(value *
value2);
}

/*Jos valitaan toinen merkki
comboboxista, sitä käytetään laskuun*/
else if (selectBox.SelectedIndex == 3)
{
    //Vastaus-labelin viereen tulostuu syötettyjen arvojen osamäärä
    answerLabel.Text = "Vastaus: " + Convert.ToString(value /
value2);
}
```

Tällöin näkymä tekee saman työn, kun valitaan muita merkkejä laskemiseen.

# 5 JOHTOPÄÄTÖKSET

Tämän työn tarkoituksena oli tutkia C#-ohjelmointikielestä laajuisesti, että mitä se sisältää ja mitä se mahdollistaa ohjelmointityössä. Sillä on lukuisia ominaisuuksia ja mahdollisuuksia, joiden kautta ohjelmoijat voivat kehittää monipuolisia projekteja, sillä sitä voidaan käyttää mm. ohjelmistotuotantoon, peliohjelmointiin, verkkokehitykseen, jne. Kieltä verrattiin myös muihin kieliin ja tutkittiin, miten tämä poikkeaa muista. Kielen ominaisuuksista kerrottiin myös siitä, miten se on rakennettu ja miten se toimii. Samalla on kerrottu tämän historiasta ja versiojulkaisuista.

C#-ohjelmointikieltä on melko yksinkertaista käyttää. Sillä on lukuisia automatisoituja toimintoja, joita muilla kielillä ei ole. Myös siihen liittyviä alustoja on kätevää käyttää saadakseen C#-ohjelmoinnin toimimaan niiden kanssa, kuten Visual Studion projekteissa. Kun verrattiin mm. C++-kieleen, C#-kielellä on yksinkertaistettu enemmän asioita, joita C++-kielellä tehdään manuaalisesti, mikä siis vähentää sen kanssa työskentelyä. Jos on oppinut käyttämään jo yhtä kieltä hyvin, niin muita kieliä on tällöin helpompaa opetella. C# on myös melko uusi muihin verrattuna, joten se tarjoaa moderneja mahdollisuuksia. Se sisältää hyvin monipuolisesti ominaisuuksia, joista merkittävimpiä ovat .NET ominaisuudet. Ensimmäiseksi pyrittiin kertomaan teoriasta ja sitten käytännöstä.

Käytännössä katsottiin läpi, miten luodaan ohjelmistoprojektia .NET 6:tta käyttäen. Tämän jälkeen lähdettiin tutustumaan, miten ohjelmointi yleisesti toimii. Tehtiin erilaisia pienohjelmia ja syötettiin tulokset ohjelmista mukaan. Katsottiin, miten luokat, ehdot, funktiot, silmukat, taulukot ja käyttöliittymät toimivat C#-kielen kanssa Visual Studiossa. Laitettiin myös koodeihin dokumentaatiota, jossa kerrottiin mitä ne tekevät. Nämä auttavat lukijaa ymmärtämään mistä on kyse. Lähdettiin aluksi tekemään ohjelmia maista, joista kysyttiin käyttäjältä vastaamaan niihin ohjelmassa. Tämän jälkeen lähdettiin tekemään käyttöliittymän puolella laskinta, joka vastaa siihen mitä käyttäjä syöttää käyttöliittymässä.

Tutkimuksessa havaitsin, miten .NET Framework toimii C#:n taustalla ja mitä se tälle mahdollistaa. Löysin samalla paljon tietoa sen historiasta ja ominaisuuksista. Opin, miten C#-kieli itsessään toimii ja miten se on rakennettu. Pääsin myös tekemään pientä ohjelmaa, josta sain myös selittää, miten se toimii. Pääsin siis tutkimuksessa hyödyntämään sekä teoriaa, että käytäntöä.

**LÄHTEET**


Anshul, A. 2022. Introduction to .NET Framework. Saatavissa: https://www.geeksforgeeks.org/introduction-to-net-framework/. Viitattu 15.10.2022.

Chand, M. 2022. What Is New In .NET 6.0. Saatavissa: https://www.c-sharpcorner.com/article/what-is-new-in-net-6-0/. Viitattu 15.11.2022.

Dwij, D. 2020. LINQ in C#. Saatavissa: https://www.c-sharpcorner.com/UploadFile/72d20e/concept-of-linq-with-C-Sharp/. Viitattu 3.10.2022.

Harkiran. 2021. .NET Framework Class Library (FCL). Saatavissa: https://www.geeksforgeeks.org/net-framework-class-library-fcl/. Viitattu 15.10.2022.

Jain, S. 2017. Overview of Common Language Infrastructure. Saatavissa: https://www.c-sharpcorner.com/blogs/overview-of-common-language-infrastructure. Viitattu 15.10.2022.

Microsoft. 2022a. .NET CLI overview. Saatavissa: https://learn.microsoft.com/en-us/dotnet/core/tools/. Viitattu 16.10.2022.

Microsoft. 2022b. .NET Framework versions and dependencies. Saatavissa: https://learn.microsoft.com/en-us/dotnet/framework/migration-guide/versions-and-dependencies. Viitattu 15.10.2022.

Microsoft. 2022c. A tour of the C# language. Saatavissa: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. Viitattu 17.11.2022.

Microsoft. 2022d. Attributes (C#). Saatavissa: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/attributes/. Viitattu 17.11.2022.

Microsoft. 2022e. Common Language Runtime (CLR) overview. Saatavissa: https://learn.microsoft.com/en-us/dotnet/standard/clr. Viitattu 3.10.2022.

Microsoft. 2022f. Desktop Guide (Windows Forms .NET). Saatavissa: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-6.0. Viitattu 17.11.2022.

Microsoft. 2022g. Ecma Standards. Saatavissa: https://learn.microsoft.com/en-us/dotnet/fundamentals/standards. Viitattu 15.10.2022.

Microsoft. 2022h. Visual Studio 2022. Saatavissa: https://visualstudio.microsoft.com/vs/. Viitattu 15.10.2022.

Microsoft. 2022i. Visual Studio Code. Saatavissa: https://code.visualstudio.com/. Viitattu 15.10.2022.

Microsoft. 2022j. What is ASP.NET?. Saatavissa: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet. Viitattu 15.11.2022.

Microsoft. 2022k. What's new in .NET 6. Saatavissa: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-6. Viitattu 17.9.2022.

Mono Project. 2022. About Mono. Saatavissa: https://www.mono-project.com/docs/about-mono/. Viitattu 15.10.2022.

Setia, V. 2020. Understanding .NET Framework, .NET Core, .NET Standard And Future .NET. Saatavissa: https://www.c-sharpcorner.com/blogs/understanding-net-framework-net-core-and-net-standard-and-future-net. Viitattu 15.10.2022.