Sommarlovs studier 2021

Stödord:

Abstraktion: Kod eller teknik som är enkel att använda utan att veta hur det fungerar.

Active record: Information från en databas som visas upp för en användare.

Agile programutveckling: Är en arbetsprocess för att utveckla program.

AJAX: En metod för att hämta data från en webbserver som använder sig av XML, JS och HTML.  
  
Algoritm: Kan t.ex vara roten ur 5 eller omkretsen för en pyramid.

Angular.js: En JS front-end ramverk för att bygga hemsidor. En samling av templates och förkodad kod.

Apache: Apache eller Apache http server är en open source gratis webb program. Haft stor betydelse för växten i internet och är A i LAMP stack.

API: Applikations programmerings gränssnitt som tillåter interaktioner mellan flertal mjukvaru program så att utvecklare kan hämta information från extern mjukvara. Googles api tillåter användare att hämta bilder och mappar till deras egna program.

Apprenticeship: En överenskommelse mellan arbetsgivaren och den anställda. Där den anställda får träning och betalar i utbytte mot arbete.

Argument: Ett nummer, text eller liknande som är nödvändigt för att en funktion ska kunna köras.  
  
Arithmetic Operators: Dessa operatörer används med nummer för att utföra enkel matte, t.ex. addition. Datorer måste kunna detta för att ens kunna utföra något.

Array: Som en buss där det finns ett antal säten som antingen är fyllda med ett värde eller är NULL.

ASCII: American Standard Code är ett system för elektronisk kommunikation. Som har 128 nummer där varje nummer står för ett tecken eller symboler. Är samma över hela världen.

Assignment Operators: En operator som anger ett värde till en variabel, kan vara =.

Asynchronus learning: Lärande som kan ta plats olika beroende på student, matrialet är ofta förinspelat. T.ex Pluralsight \*usch\*.

Asyncronus Programming Language: Ett programmeringsspråk som kan utföra flertal saker i stunden och går inte i ordning.  
  
Augmented Reality: Mjukvara som sätter digitala objekt på bilder och videor. Kan vara att din mobil känner av att du ska ta ett porträtt av dig själv eller bild på din hund och justerar bilden därifrån.

Autonomous: Själv guidande och kapabel att arbeta helt för sig själv utan respons från en annan. Många drönare och bilar är det.

Back End: Den sidan av programmet som du inte kan se, där alla uträckningar sker osv.

Backbone.js: JS bibliotek som man mestadels använder sig av till en sid web appar för att ge struktur och hantera användar inmatning och interaktioner.

Binary: System av två möjliga värden, 0 eller 1. Datorer använder sig av detta och betyder att de samlar och räknar ut saker i nollor och ettor.

Binära alfabetet: 0 eller 1

Binära nummer: En samling av nollor och ettor som utgör ett program.

Bit: En nolla eller etta. Minsta informationen en dator kan ta emot.

Block-based Programming Language: Ett visuellt programmeringsspråk. Block-based tillåter använder att dra och tappa block av kod för att skapa program. Scratch exempel.

Boolean: Sant eller falskt.

Bootstrap: En open-source ramverk. En grupp av templates för att bygga front enden på en hemsida. Utgörs av stora filer av HTML, CSS och JS.  
  
Bug: Kod som är gjord på ett sätt som gör att program avslutas eller inte går att köras.

Build: Bygga program är att göra det användbart av användare. Kodning, testning och debug för att bli klart.

Byte: Består av 8 bitar. Alltså åtta noller eller ettor blandat.

Kapitel 1:

.Net Framework är en utvecklingsplatfrom som inkluderar Common Language Runtime(CLR), som utför körningen av kod. Och ett Base Class Library (BCL) som är ett rikt bibliotek av klasser.  
  
LTS står för Long Term Support.

Windows forms och Windows Presentation Foundation (WPF) kan använddas för att bygga graphical user interface (GUI).  
  
ASP.NET Web Forms och Windows Communication Foundation(WCF) har blivit ersatt av ASP.NET MVC och ASP.NET Web Api. Dessa två har blivit omgjorda och kombinerade till en platform som körs på .Net 5, kallad ASP.NET Core.  
  
EF6 är en object-relational mapping (ORM) som är designad att fungera i relations databaser som Oracle och Microsoft SQL Server. Och heter nu Entity Framework Core.

.Net Core: För cross platform och nya appar.

.Net Framework för legacy apps.

Xamarin för mobila appar.  
  
Dessa sammanlades till .Net Standard för att förebygga krånglighet. Inbyggt i alla tre ovanför.

C# compiler, kallad Roslyn omvandlar C# source code till intermidate language(IL) är en assembly av DLL eller EXE fil. Som sedan utfärdas av .NETs virtuella machine kallad CoreCLR. Vid körning laddar CodeCLR, IL koden sedan kompilerar just-in-time (JIT) det till maskinkod som sedan utfärdas av datorns CPU.

**Exercise 1.1**

1. För att .Net är ett ramverk och inte själva kodningsspråket.
2. dotnet new console
3. dotnet run
4. ctrl+shift+ö
5. För att bara använda Windows, ja. Vill man köra .NET på Linux och MacOS så är VSC det som gäller.
6. .Net Framework är inte open source och används främst för att bygga windows applikationer. .Net Core är open source och funkar på Linux och MacOS och är bättre om man ska utveckla saker som ej är låsta till Windows.
7. .Net Standard är det som sammanför .Net Core, .Net Framework och Xamarin vilket gör att det inte blir lika knöligt att använda sig av alla tre.
8. static void Main()
9. På Microsofts docs
10. Stackoverflow och microsoft docs.

**Kapitel 2**

Subtotal är en variabel medan subtotal + salestax är ett uttryck.

Opeator är t.ex. + eller -. Operanderna är de som används som värden mellan operatorerna. Block är {}, de kan innefatta klasser, namespaces och metoder eller ett statement. C# vocabularyn är uppbyggd av keywords, symbol characters och typer.  
  
Bil och hund är typer, Huvud och motor är fält som är nouns till typerna. Medan Fido och Bob är variabler. Nykelord som verkar vara typer är aliaser. Typer kan förväxlas med klasser. 64bit kan vara snabbare 16bit variabler då koden kanske körs på en 64bitars maskin. nameof kan användas för att skriva ut namnet på en variabel. Literal värden är värden som passar datatypen.

42 hela nummer, -42 int, riktiga nummer 3.9. Double 0.1 är inte 0.1. Decimal är mer pålitlig då den lägger 0.1 som 1. Double infinity, epsilon och nan.

Intellisense kan ej läsa av dynamic, istället gör CLR vid runtime och kastar ett exception om det saknas. Använda var för att förminska kod skrivning, då man kan se värdet på den i new statmentet.

String är en referenstyp och innehåller ett minne av värdet och är själv inte värdet. Null är default värdet för alla referenstyper.

Interpolated strings är strängar som börjar med ett $. N0 nummer med tusen separatorer och inga komma tecken. C står för currency.

Fulla formatet för ett format item: {index [, alignment] [ : formatString]}

**Exercise 2.1**

1. Ulong
2. Byte
3. Byte
4. Ulong
5. Uint
6. Ushort
7. Byte
8. Ulong
9. ulong
10. ushort

**Kapitel 3**

Använd aldrig ++ och -- operatorn med assignment operatorn =.

Att byta bits är snabbare än att multiplicera.

& är när båda talen har en gemensam 1 t.ex båda 0010

| är vilka bites som är 1 t.ex ena 0100 och andra 0011

^ är när ena biten är ett på ena talet men 0 på det andra

Nameof returnerar namnet på variabeln och sizeof returnerar värdet på den. Punkten efter en variabel och dess medlemmar kallas member access operators och bracketsen efter en funktion eller metod kallas invocation operator.

Switch uttryck använder sig av lambda för att returna. Implicit casting är säkert och görs automatiskt. Explicit casting är inte säker och du kan förlora information. Casting tar bort decimalen medans casting avrundar. Rundar upp om talet är udda t.ex 13.5, men neråt om talet jämnt t.ex 12.5. Använd midpointrounding om man alltid ska runda upp från 0.5.

**Exercise 3.1**

1. Den throwar ett DivideByZeroException
2. Värdet gick från 5.5 till 8, alltså en ökning med 2.5
3. Om programmet inte körts så säger kompilatorn att värdet är för stort, om programmet körs så throwar den ett overflowexception.
4. y++ utförs efter assignment, alltså efter kodsnuten körts. ++y körs på direkten.
5. Break avslutar kodstycket, t.ex en switch sats. Return hoppar ur metoden, ofta med ett värde. Continue fortsätter från där den är, t.ex i en for loop så hoppar den ur och fortsätter koden.
6. Initializer, condition och iterator. Bara initiazlier och condition.
7. = är assignment operator och sätter ett värde på t.ex. en variabel. == jämför två värden och returnerar sedan sant eller falskt.
8. Nej, det gör det inte.
9. Default värdet i en switch sats.
10. IEnumerable eller enumerable. Förmodligen bara IEnumerable.

Kapitel 4

Nummer som används för att räkna kallas cardinal numbers. Medan nummer som används för t.ex första, andra och tredje kallas ordinal numbers. Factorials är t.ex 5x4x3x2x1 = 120. Så factorialen av 5 är 120. De skrivs med 5! och kallas bang. Recursion är när en funktion kallar på sig själv. Använd checked för att hålla koll på overflowexceptions. /// för bättre beskrivning av funktioner. Du kan ändra värden medans du debuggar.  
  
Trace listener är en typ som kan justeras till att skriva utmatning. Debug används för att logga under kodens skrivning utveckling. Trace används för att logga under kodens utveckling och runtime.

Debug writeline stryks helt från koden när den inte kör i debug. TDD aka Test-Driven Developement, är när man hittar buggar innan man skrivit koden.

En väl skriven unit test har 3 punkter: 1. Arrange där man deklarerar och instanserar variabler för in och utmatning. 2. Act, delen där du utför testning. I vårt fel är det metoden som vi kallar på för att testa. 3. Den här delen är hyptoser om vad som kommer hända i programmet t.ex om vi tror 2\*6 blir 12.

**Exercise 4.1**

1. Void betyder att inget returneras, någonting händer i funktionen men inget kommer tillbaka.
2. Imperativ programmering är när man använder vissa statements för att uppnå något t.ex med variabler och operators. Funktionell programmering är när metoder gör uträkningarna som exempel.
3. Om man ska köra med eller utan debuggern.
4. Output build och även en fil om man ställt in det.
5. 0 off, 1 Error, 2 Warning, 3 Info och 4 Verbose.
6. Trace används för att spåra vad som händer i en kod och försvinner ej vid färdigt program medan när programmet är färdig byggt så finns inte t.ex debug.writeline kvar.
7. Arrange, act och assert.
8. [Fact]
9. Dotnet test.
10. Test driven developement som man använder för att hitta buggar innan man skrivit programmet.

**Kapitel 5**

Enkapsulation är kombinationen av data och actions som är relaterade till ett objekt. Där du bestämmer vad som kan komma åt vad.

Composition är vad ett objekt är gjort av. T.ex namn, ålder osv.

Aggregation är vad som kan kombineras med ett objekt, t.ex en bil som har objekt av människor.

Inheritance är när man ärver saker från en annan klass.

Abstraction är grundiden av ett objekt som ignorerar detaljerna. Bar concreta klasser kan skapa objekt medan abstracta inte kan.

Polymorphism är att tillåta derived classes att skriva över en ärvd funktion.

Access modifier är de som tillåter vilka som får använda klasserna och funktionerna.

Fields används för att lagra data. Finns 3 olika sorter. 1. Constant som aldrig kan ändras. 2. Read-only datan kan inte ändras efter att klassen är instanserad. 3. Event något som du vill ändras när något händer.

Methods används för att köra statements. Constructor: Statementsen körs när du instanserar en ny klass. Property är statements som körs när du hämtar eller lämnar data. Lagras vanligen i ett fält eller kalkuleras vid körning. Vanligen så man inkapsulerar fält. Indexer är när du hämtar eller lämnar data genom array syntax. Operator som + eller -. Dessa är 4 specialiserade metoder och kategorier inom metoder. ”Alla” klasser ärver från System.Object som innehåller ToString(). Klassen man ärver mot är base eller superklassen medan den som ärver kallas dervied eller subklass.

Access Modifiers: Private: Membern är tillgänglig i typen bara, default. Internal: Tillgängligt i typen och i samma assembly. Protected: Tillgängligt i typen och vilken typ som ärver från den. Public: Tillgängligt överallt.  
  
Statiska metoder som alltid kommer vara samma i princip, instansering vid nya objekt som är speciella. Använd read only fält över const, för att värdet kan bli kalkulerat eller laddat under runtime och uttryckt. Konstruktor körs innan värdet av fält sätts.

Tuples kan ta namn från andra tuples så istället för t.ex item1 och item2 så kan den ta namn och nummer från en annan tuples i en klass.

Utan att ange ett namn på tuplesen så kan du skapa nya variabler på returvärdet från funktionen/metoden. Kallas deconstruction. En metod signatur är en lista av parametrar som kan passas när man kallar en metod.

När man skickar med ett värde har det samma värde efteråt, skickar man med ref värde så ändras det ursprunliga värdet, out värdet skrivs över helt, skapas på nytt.

Skillnad på \_ och utan? Init gör så att du bara kan instansera en property en gång. Med records skapar du nya värden från de gamla, detta är non-destructive mutation.

**Exercise 5.1**

1. Public alla har tillgång till det, private bara den egna klassen har tillgång till den, internal bara klasser i assembliet har tillgång till den, protected klasser som ärver från den.
2. Readonly har bara ett värde från börjat, const har alltid samma värde, static är att den inte behöver instanseras kan vara en allmän metod.
3. I konstruktorn ger du instanseringen världen som exempel.
4. När det kan innehålla fler värden t.ex Sverige har blått och gult därför blir det blå | gul.
5. För att förenkla klasser, ifall man behöver dela upp det för enkelhetensskull.
6. En ”variabel” där du kan kan lagra olika värden i.
7. Det tillåter dig att återanvända en variabel. Kopiera den och ändra på innehållet.
8. Att du kan ha olika antal parametrar och att du då kan anpassa det specifikt för den metoden eller konstruktorn.
9. Fältet är där du lagra värden, i propertyn bestämmer du hur du ska hämta och ge värden.
10. Genom att sätta ett default värde.

Kapitel 6

Instans objekt är actions som ett objekt gör mot sig själv. En statisk metod är actions som typen gör. En metod som skapar ett nytt objekt eller ändrar på ett existerande ska returnera ett resultat till användaren av det nya objektet. Du kan skapa metoder med operatorer. Lokala funktioner inuti en klass kallas ibland nested eller inner functions.

En delegate innehåller en minnesadress med samma signatur som metoden. Om du bara ska ha en metod lagd i delegate fält så använd ej event i delegaten och bara =, om det ska finnas fleras metoder använd event och +=

Interface som ärvs får bara sakna en grej. Generics tillåter att man skickar in typer som parametrar. Generica metoder behöver man inte ange typen, det listar kompilatorn ut själv.  
  
Det finns 2 minnestyper, stackminne och heapminne. Båda två kan likaväl vara fysiska som virtuella. Stackminnet är snabbare då det hanteras av CPU:n på direkten. Heapminne är långsammare men finns gott om. Det finns 2 C# keywords för att skapa object typer: klasser och struktar. Båda är i princip likadana förutom hur de allokerar minne. När du definerar en klass så använder du en referens typ. Det betyder att minnet som objektet själv använder lägger sig på heapen, och bara minnes adress(och lite över) lägger sig på stacken.

När du definerar en strukt så definerar du en värdetyp, vilket betyder att datan för det objektet lägger sig på både stacken och heapen.

Struct Types:

Numbers: byte, sbyte, ushort, short, int, uint, long, ulong, float, double och decimal.

Miscellaneous: char, DateTime och bool

System.Drawing: Color, Point och Rectangle

Nästan resten av alla typer är klasstyper, inklusive string.

Förutom skillnaden på var minnet läggs så kan du ej ärva från en strukt.

Destructor är när man förstör ett objekt och deconstructor är när man tar ut delar från ett objekt. New kan användas på en ärvd metod med samma namn för att visa för kompilatorn att den ska ersättas. Med sealed keyword kan du motverka att din klass ärvs. När man använder new keywordet i en klass som ärver så heter det non-polymorphic inheritance eller om man overridar det så kallas det polymorphic inheritance. Om metoden har override eller virtual i sig så vet kompilatorn att det är den metoden den ska använda och inte i den ärvda. Använd virtual och override istället för new.

Implicit casting är när du castar ditt objekt som ett objekt i den ärvda klassen och explicit casting är tvärtom. Sedan C#9 så kan man använda is not istället för ! inom parantes. Använd is och as för att undvika att throwa exceptions.

**Exercise** 6.1

1. En delegate är en typ som referrar efter en metod med viss inparameter.
2. Ett event är som en delegate fast du kan lägga in flera metoder i den som ska köras.
3. Base klassen är den ursprunliga klassen och derived klassen är klassen som ärver från den ursprungliga klassen. Man kommer åt Base klassen i derived klassen med keywordet Base.
4. Is operatorn returnerar sant eller falskt när man jämför medans as returnerar objektet eller null.
5. Sealed
6. Static
7. Override
8. Destructor är när man förstör ett objekt och deconstructor är när man tar ut delar från ett objekt.
9. Exception
10. Extension method är en statisk metod som tar in ett eller flera värden. Det defineras med this som inparamter.

Kapitel 7

.Net är uppgjort av flera delar.

Language compilers dessa gör din skrivna kod (C#, F# och Visual Basic) till intermediate language (IL) lagrad i assemblies. Efter C# bytte Microsoft till en open source rewritten compiler som heter Roslyn som Visual Basic också använder sig av.

Common Language Runtime(CoreCLR): Denna runtimen laddar assemblies, kompilerar IL koden till native code instruktioner till CPU:n och kör koden inom miljön som hanterar trådar och minnen osv.

Base Class Libraries(BCLs) av assemblies i NuGet Packages(CoreFX) dessa är förbyggda assemblies av typer packade och distruberade med NuGet för att utföra vanliga uppgifter.

En assembly är en typ som är lagrad i filsystemet. Assemblies är en mekanism för att använda data. T.ex. System.Data, den måste också ha en referens. De är oftast distruberade genom NuGet packages. Project SDKs och platforms är kombinationer av NuGet paket.

Namespace är en adress av en typ. DLL står för dynamic link library. Om du behöver en cirkulär referens, använd interface.

XDocument är en typ av namespacet. System.XML.Linq. Använd keywords istället för typer när det går. Innan .Net Standard fanns Portable Class Libraries(PCLs), med den kunde du specifera vilken plattform som biblioteket skulle supporta. Nu finns .Net Standard, en API som alla .Net plattformar har stöd för. Använd .Net Standard 2 istället för 2.1.

Tre sätt att använda dina applications för deployment: 1. Framework-dependent deployment (FDD). 2. Framework-dependent executables (FDEs) och Self-contained. För att minska på den publishade koden så kan man använda <PublishTrimmed>true</PublishTrimmed> och <TrimMode>Link</TrimMode>. Undvik addtional qualifiers som betas (beta1), release candidates (rc4) och wildcards (\*).

**Exercise 7.1**

1. Namespace är en address av en typ och en assembly är en typ lagrad i filsystemet.
2. Add reference
3. Det kan dekompilera b.la exe och DLL filer. Det kan också ge ett hum om hur ”professionella utvecklare” skriver sin kod.
4. System.Single
5. Package Manager Console för att restora.
6. Framework-dependent kräver att man har installerat Framework sedan tidigare medan self-contained har allt redo fast tar mer plats.
7. Är värdet som symboliserar vilken plattform och version som koden kan köras på när man publishar.
8. Pack gör ett NugetPackage av din kod. Publish gör att din kod kan köras direkt på en annan maskin beroende på hur den är publicerad, ish.
9. På Windows, Linux och MacOS: ASP.Net Core MVC/Web Api och Konsolapplikationer. Även på Windows: Windows Forms Applications, WPF och UWP. Funkar dock ej med ASP.Net Web Forms, Windows Communication Foundation och Silverlight.
10. Ja, det kan du oftast.