Laboration 1 Mikael Eriksson

UPPGIFT 1

Uppgift 1A Räkna A

1.

Planering

Steg 1	Skapa en tom konsolapplikation i Visual Studio.
Steg 2	Skapa lämpliga variabler.
Steg 3	Göra så att användaren kan mata in en textsträng.
Steg 4	Använda en for-loop för att gå igenom bokstäverna en och en.
Steg 5	Använda if-satser för att ta reda på om tecknet är ett ”a” eller ”A” och sedan addera 1 till variabeln som håller ordning på antal a.
Steg 6	Skriva ut resultatet.

Planerad tidsåtgång: 1h.

2.
a.
Steg 1 och 2 tog max 5 minuter eftersom jag har gjort det så många gånger förut. På steg 3 uppstod inte heller några problem. Den största delen av tiden ägnade jag åt steg 4 där jag stötte på vissa hinder.

Tidsåtgång: 1h 15 min.

b.
Problemet jag stötte på var att det inte i if-satsen gick att jämföra det aktuella tecknet med bokstaven a. Jag skrev ”if (line[i] == ”a”), vilket inte fungerade eftersom det är olika typer – string och char. Till slut provade jag att skriva a:et med enkla citattecken: ’a’, vilket fungerade. Att det gick att använda egenskapen length och loopa igenom textsträngar förstod jag efter att ha läst ett exempel från kursen ”Inledande programmering med C#” med titeln ”WhatAscii”.

c.
Problemet med olika datatyper i if-satserna ledde till avvikelse i planeringen.

3.
På den här uppgiften visste jag ungefär hur jag skulle göra innan jag satte igång och kunde därför skriva en grundläggande planering. För att komma på exakt hur jag skulle göra var jag tvungen att gå tillbaka till lite av det material vi hade på kursen i C#. Jag kollade också efter lösningar på nätet eftersom min if-sats inte fungerade på grund av att det var olika typer. Det visade sig att lösningen på problemet var enklare än jag trodde. Att söka på nätet efter lösningar kan vara ganska stort slöseri med tid. Det är svårt att finna något som är tillämpbart på ens uppgift och ofta är lösningar som presenteras på forum och liknande onödigt omständliga.

Uppgift 1b Räkna Siffror

1.

Planering

Steg 1	Skapa en tom konsolapplikation i Visual Studio.
Steg 2	Skapa variabler för heltalet, samt för antal nollor, udda och jämna siffror.
Steg 3	Använda en for-loop för att gå igenom talet siffra för siffra.
Steg 4	Använda if-satser och %-operator för att addera till lämplig variabel.
Steg 5	Skriva ut resultatet.

Planerad tidsåtgång: 30 min.

2.
a.
Uppgiften tog nästan exakt en halvtimme enligt planeringen. Jag stötte på några problem men klarade av att lösa dem ganska snabbt. Mest tid tog if-satserna och frågan om vilka datatyper som var lämpliga att använda.

Tidsåtgång: 30 min.

b.
Jag stötte på hinder först på tredje momentet där jag skulle loopa igenom numret siffra för siffra. Jag trodde att man kunde använda egenskapen length eller motsvarande på int. Eftersom det inte var möjligt omvandlade jag numret till en sträng. Problemet som uppstod då var att det inte går att göra uträkningar på en sträng. Därför fick jag i if-satsen utföra en explicit typomvandling. Det hade varit möjligt att läsa in numret som en sträng redan från början, dvs inte konvertera med int.Parse, men då hade det varit svårare att kontrollera om användaren verkligen matat in ett heltal.

c.
Jag hade inte direkt några avvikelser från planeringen.

3.
Tack vare att jag hade gjort uppgift 1a visste jag att uppgiften skulle gå att lösa genom att omvandla heltalet till en sträng. Annars är det inte säkert att det hade gått så snabbt. Jag kan även tillägga att jag fick för mig att man inte fick använda arrayer i uppgift 1a och 1b.

Uppgift 1c Näst största

1.

Planering

Steg 1	Skapa tom konsolapplikation i Visual Studio.
Steg 2	Skapa nödvändiga variabler.
Steg 3	Skriva ut instruktioner till användaren.
Steg 4	Låta användaren skriva in 10 stycken heltal.
Steg 5	På något sätt jämföra de inlästa talen med varandra.
Steg 6	Skriva ut resultatet.

Planerad tidsåtgång: 2h.

2.
a.
Uppgiften tog längre än de planerade två timmarna. Bara att komma på hur heltalen skulle skrivas in tog nästan en timme. Min tanke var att alla talen skulle läsas in och först därefter skulle programmet undersöka vilket som var det näst största. Det hade gått om man fått använda en array. Efter att ha provat det fick jag överge min första plan till något som mer liknade:

Steg 1	Användaren får bestämma hur många tal som ska läsas in.
Steg 2	Inläsningen placeras i en while-loop som körs så många gånger som användaren har bestämt.
Steg 3	Med hjälp av if-satser avgör jag om det inlästa talet är större eller mindre än det tidigare inlästa.

Tidsåtgång: 2h 45 min.

b.
Det största felet jag gjorde var att försöka läsa in alla talen på en gång och sedan gå igenom dem. När jag sedan förstod att jag var tvungen att med hjälp av if-satser jämföra det senast inlästa talet med de tidigare talen placerade jag if-satserna fel. Jag hade ingen exakt plan över hur if-satserna skulle placeras och vilka villkor de skulle innehålla. Därför fick jag pröva mig fram. Jag kompilerade koden, ändrade, kompilerade igen, och så vidare tills jag fick önskat resultat.

c.
Anledningen till avvikelsen i planen har jag redan varit inne på. Problemet var att jag inte visste hur jag skulle göra från början och fastnade i fel spår istället för att överge det och börja på ett nytt angreppssätt. Det tog alltså lång tid innan jag förstod att det inte skulle fungera.

3.
Den här uppgiften var så pass svår att jag inte från början visste exakt hur den skulle skrivas. Jag hade en ungefärlig plan som jag fick överge för att den visade sig inte fungera. Jag var tvungen att tänka på ett annat sätt och insåg att jag inte behövde spara alla de inmatade talen. Efter att ha spanat på ett exempel från C#-kursen där användaren fick bestämma hur många gånger ett meddelande skulle skrivas ut, insåg jag hur en lösning på mitt problem skulle kunna utformas. Tyvärr är det

tidskrävande att ge sig in i ett problem utan att ha någon tydlig planering för hur det ska lösas. Hade jag gått igenom exempel och möjliga lösningar innan hade uppgiften förmodligen kunnat lösas mycket snabbare.

Det tog även lång tid att placera if-satserna och utforma dess villkor så att det gav önskat resultat. Jag vet fortfarande inte om de är skrivna på det bästa möjliga sättet. Jag testar mig fram och flyttar på if-satserna tills det fungerar utan att riktigt förstå hur. Här är det förmodligen bättre att sätta sig ner och rita upp en algoritm som beskriver hur lösningen ska se ut.

UPPGIFT 2

2a.
Genom att reflektera över mitt eget arbete har jag insett att jag behöver en tydligare plan redan från början, men samtidigt kunna ändra den om det inte skulle fungera. På svårare uppgifter vet jag inte exakt hur jag ska göra när jag sätter igång och har därför en väldigt skissartad plan. Efter hand växer i bästa fall förståelsen och jag kommer på lösningar som gör att jag tar mig vidare. I värsta fall fastnar jag och försöker mig på lösningar som inte kommer att fungera. Jag brukar köra på tills jag fastnar och tvingas söka efter en lösning genom att kolla på kursmaterialet, tidigare föreläsningar eller söka på nätet. Ofta är det rätta svaret mycket enklare än man trodde. Kanske är det bättre att läsa på innan för att få en tydligare planering och undvika de vägar som kan visa sig vara mycket tidskrävande att ta sig ur.

Jag ägnar också mycket tid år att en slags try-and-fail ansats som också kan vara tidskrävande. Jag ändrar och kompilerar tills det fungerar utan att riktigt förstå varför. Kanske är det bättre att skriva ner en sorts algoritm eller bli bättre på att i VS stega igenom koden för att se och förstå vad det är som händer. Jag talar här till exempel om if-satsernas placering i uppgift 1c.

2b.
Jag tror att det går att minska konsekvenserna av fel och annat som inträffar genom att ha en tydligare plan från början och framför allt läsa på innan man börjar programmera istället för att söka efter lösningar först när man kört fast. Samtidigt är det svårt att veta exakt hur man ska göra när det är svårare uppgifter. Man måste få prova sig fram och se vad som skulle kunna fungera. Sedan krävs det så klart erfarenhet: om man har löst en liknande uppgift tidigare är det mycket enklare att ta sig an den nya.

2c.
1. Tydligare planering innan jag börjar där jag delar in uppgiften i mindre delar.
2. Försöka läsa på och fundera över möjliga lösningar på problem som kan uppstå innan jag börjar programmera.

UPPGIFT 3

Uppgift 3a Palindrom

1.

Planering

Steg 1	Jag ska börja med att söka efter möjliga lösningar, liknande exempel från kursmaterial etc.
Steg 2	Med hjälp av en for-loop som läser textsträngen baklänges ska en ny textsträng skapas.
Steg 3	En ny textsträng ska alltså byggas. Jag måste undersöka hur man gör detta.
Steg 4	Använder sedan if-satser för att jämföra de båda strängarna med varandra.
Steg 5	Sista steget är att presentera resultatet.

Planerad tidsåtgång: 1h.

2.
a.
Jag hittade ett exempel ur kursmaterialet på ett program som skriver ut alfabetet baklänges och kunde använda en liknande metod för att lösa mitt problem. Därför tog det inte lång tid att lösa hur själva for-loopen skulle utformas. Det som tog tid var att komma på hur den nya strängen skulle byggas. Sedan upptäckte jag att programmet inte fungerade på meningar. Det tog tid att komma på hur man gör för att radera blanksteg.

Tidsåtgång: ca 1h 15 min.

b.
Programmet fungerade inte på meningar eftersom jag inte använde rätt metod för att ta bort mellanslag. Jag försökte med metoden string.Trim, men upptäckte att den enbart tar bort mellanrum före eller efter strängen. Sedan upptäckte jag att det gick att använda metoden replace. Jag visste inte heller från början hur jag skulle addera bokstäver till den nya strängen. Försökte med StringBuilder och liknande men upptäckte att det helt enkelt gick att använda += inuti for-loopen för att addera bokstäver till den nya strängen.

c.
Jag hade inte några större avvikelser från planeringen. De jag hade beskriver jag ovan.

3.
På den här uppgiften försökte jag ha en tydligare plan för hur jag skulle göra innan jag började. Bland annat försökte jag hitta lösningar på liknande problem i kursmaterialet vilket gjorde att jag visste hur man loopade igenom en sträng från slutet till början. Jag visste sedan hur jag skulle göra för att jämföra de båda strängarna med varandra. Hur den nya strängen skulle skapas visste jag på ett ungefär, men det hade jag kunnat ta reda på innan också. Att programmet inte skulle fungera på meningar var inget jag tänkt på innan utan det var ett oförutsett problem som jag fick hantera under tiden.

Uppgift 3b Bråktal

1.

Planering

Steg 1	Försöka förstå instruktionen, dvs. vad uppgiften egentligen går ut på och ställer för krav.
Steg 2	Skapa en ny klass och lägga till de medlemmar den ska innehålla.
Steg 3	Skapa egenskaper och en konstruktor som tar två argument.
Steg 4	Testa att konstruktorn fungerar genom att skapa ett nytt Fraction-objekt och skicka med en täljare och nämnare som argument.
Steg 5	Fundera över hur man ska kunna jämföra och göra uträkningar på två Fraction-objekt.
Steg 6	Arbeta med de andra medlemmarna i klassen Fraction.
Steg 7	Testa genom att mata in olika bråktal. Ändra tills jag får önskat resultat.
Steg 8	Förbättra genom att exempelvis lägga till en metod som förenklar bråk.

Hade ingen planerad tidsåtgång på denna uppgift.

2.
a.
Jag arbetade på den här uppgiften under flera dagar. Sammanlagt tog det ungefär 5 h. Eftersom instruktionen är så otydlig (kanske medvetet) tog det lång tid innan jag förstod vad uppgiften egentligen gick ut på. Har ännu inte riktigt förstått det. När jag sedan bestämt mig för hur programmet skulle se ut flöt det på ganska bra.

b.
Jag förstod inte hur klassen och dess medlemmar skulle struktureras. Vad som menades med metoderna getNumerator och getDenominator kunde jag inte lista ut. Till exempel skriver man väl inte metoder i C# med inledande gemen? Jag gjorde dem sedan till Egenskaper som returnerade privata fält. Jag gjorde också fel när jag skulle jämföra och göra uträkningar på två Fraction-objekt. Försökte skapa dem inuti klassen. Kom sedan på att jag kunde ta det andra Fraction-objektet som argument till det första bråktalet.

c.
Avvikelser från planeringen berodde på att jag gjorde det extra omständligt för mig själv. När jag gjorde så som vi hade lärt oss i C#-kursen med privata fält och getter- och setter-metoder gick det mycket enklare.

3.
På såhär svåra uppgifter tycker jag att det är svårt att ha en tydlig plan framför sig när man börjar programmera. Det gäller i viss mån att börja med det man förstå och kan, till exempel skapa de medlemmar klassen ska innehålla och därefter skapa en nytt objekt och skriva ut det till konsolen. Därefter känns det enklare att gå vidare till nästa steg. Även om jag tidigare inte haft någon nedskriven plan har jag gjort mina uppgifter på det här sättet. Nu i efterhand känns det som om jag kunde ha ägnat mer tid åt att sätta mig in i problemet och utforma en tydligare planering. Men det är lätt att säga efteråt.

UPPGIFT 4

4a.
a.
Det är flera uppgifter som måste utföras. Först och främst behövs det en användardatabas som gör att det går att skapa nya användare samt ta bort användare. Det behövs även någon form av filhantering där användaren kan ladda upp och radera sina dokument. Båda dessa måste kommunicera med en hanterare av rättigheter och åtkomster som håller reda på till exempel vem som är administratör och vem som har rättigheter att se och ändra ett dokument. Det går exempelvis inte att bara radera ett dokument utan att detta går genom samlingen av användare samt dokumentets åtkomstkontrollista. Det måste också finnas funktionalitet som har med delning av dokument att göra. Såklart måste det också skapas en textredigerare med alla funktioner som den ska innehålla. En uppgift består också i applikationens utseende och gränssnitt, något som framförallt görs med css och html.

b.
Jag har ingen aning om hur lång tid det skulle ta att utföra ett sådant projekt.

Om vi säger att det tar 5 månader. Så delar man upp det i veckolånga iterationer. Första perioden går ut på att få ett igång ett system med grundläggande funktioner. Databas, server och filhantering, versionshantering och se till att de fungerar tillsammans.

Jag tror att det är viktigt att inte till exempel ägna första veckan åt användardatabasen, andra veckan åt filhantering, tredje åt textredigeraren osv, och sedan sätta ihop de olika delarna i slutet, utan utvecklingen av de olika uppgifterna måste ske parallellt. Först när de fungerar tillsammans kan man påbörja nästa steg.

c.
Jag tror att det är bra medlemmarna har bra koll på alla delarna men kanske är specialiserad på något enskilt. I så fall behöver man till exempel någon som kan det här med databashantering och någon som ägnar sig och programmering av själva textredigeraren. Sedan behövs det också någon som har ett öga för design, som ägnar sig åt att få applikationen att bli inbjudande och användarvänlig. Det krävs också att det finns någon som leder projektet, som sätter upp en planering och utvärderar arbetet, samt kommunicerar med kunden. Det viktiga – tror jag – är att de olika uppgifterna inte utförs isolerat, detta för att de olika delarna är så beroende av varandra. Därför är det viktigt att det sker en god kommunikation mellan medlemmarna av gruppen.

4.
Svårigheterna med att planera ett projekt som detta är att det är ganska omfattade och är beroende av många olika faktorer. Till exempel är det svårt att avgöra hur lång tid det kommer att ta. Om det hade varit ett företag med många år och många projekt bakom sig hade det varit enklare att uppskatta hur lång tid ett projekt av den här omfattningen skulle ta, genom att helt enkelt jämföra det med tidigare projekt. Ändå gäller det – tror jag – att sätta en deadline, kanske genom att i gruppen diskutera och enas om ett rimligt tidsperspektiv.

Sedan gäller det att sätta upp en långsiktig projektplan som inte beskriver i detalj vad gruppen eller individerna ska göra utan hur det som ska levereras ska byggas och hur det ska levereras i tid. Därefter kan man planera in exempelvis veckolånga iterationer som mer i detalj presenterar vad som ska utföras i gruppen. Iterationerna kan sedan – om man utgår från Unified process – delas in i mikroinkrement som talar om vad den enskilde individen ska göra. Innan man sedan börjar med nästa iteration är det lämpligt att projektledaren går igenom det som gjort och det som inte klarats

av för att se om projektplanen eller iterationsplaneringen behöver förändras.

Jag tänker mig att riskerna med ett sådant här projekt är att det precis som de flesta applikationer har olika delar som är beroende av varandra för att fungera. Till exempel om ett dokument ska raderas måste det ses igenom vilka andra som har rättigheter till samma dokument. Därför kan det vara av värde att se till att de stora byggstenarna står stadigt redan från början. Första steget kanske är att se till att de olika delarna kommunicerar med varandra: Gränssnitt med databasen, textredigerare med åtkomstkontrollista osv. Därefter, när applikationen har den grundläggande funktionaliteten klar, kan man ge sig på de mer avancerade sakerna och därmed närma sig det slutgiltiga målet.

Det är ungefär samma princip som gäller på mina små uppgifter. Det gäller att dela upp uppgiften i små steg och börja med det mest grundläggande och det man kan: skapa klasser, skapa ett objekt och se till att det kommunicerar med klassen. Det är ett sätt att eliminera risker, för om inte de basala funktionerna beter sig som de ska kommer det att skapa stora problem längre fram.

Källa: Andra föreläsningen.

Tidslogg

Datum	Uppgift	Ungefärlig tidsåtgång
2014-11-11	Fullständiga lösningar av 1a. Räkna A, 1b Räkna siffror och 1c Näst största.	5h
2014-11-18	Klar med uppgift 3a Palindrom och började med uppgift 3b Bråktal.	3h
2014-11-19	Fortsatte med uppgift 3b.	2h 30 min
2014-11-22	Gjorde vissa förbättringar på uppgift 3b.	30 min.