Hemtentamen,
Grundläggande programmering med C++, 2022

Uppgift 1:

a) Ett programmeringsparadigm är en grundläggande filosofi för hur man skriver program. Ett programmeringsparadigm berättar hur en person går till väga för att lösa ett problem.

b) Det procedurella paradigmet kännetecknas av tre byggstenar. Iteration, selektion och sekvens. Paradigmet är bra på att bryta ner problemet i mindre delar, som sedan blir hanterbara delproblem. Denna typ av programmering används ofta för uppgifter som måste slutföras i en specifik ordning, kan jämföras med till exempel att laga mat eller montera ihop en leksak.

   Det kan vara lättare att förstå och följa än andra programmeringsparadigm. Det kan vara mer effektivt, eftersom programmeraren kan styra i vilken ordning instruktionerna utförs. Det kan vara lättare att felsöka, eftersom programmeraren kan spåra kodens exekveringsflöde. Det kan vara mer flexibelt, eftersom programmeraren enkelt kan lägga till eller ta bort kod utan att påverka programmets övergripande struktur.

   Det kan vara svårt att förstå och följa programkoden, särskilt om den är lång eller komplex. Fel i koden kan vara svåra att hitta och sen fixa. Programmet kan vara ineffektiv, särskilt om den inte är välutvecklad. Koden kan vara svår att återanvända eller modifiera.

c) Det objektorienterade paradigmet skiljer sig från det deklarativa paradigmet genom att det fokuserar på objekt och deras relationer. Det är ett mer abstrakt och flexibelt paradigmsom gör det lättare att skapa komplexa system. Dess styrka är att det är lättare att förstå och använda, medan dess svaghet är att det är svårt att felsöka.

d) Ett problem som sannolikt skulle passa för det deklarativa paradigmet är ett problem som kräver mycket logiskt tänkande. Detta innebär att problemet kräver en mycket strukturerad och logisk lösning. Ett exempel på ett sådant problem är ett matematiskt problem. Matematiska problem kräver ofta en mycket logisk och strukturerad lösning. Detta gör det deklarativa paradigmet lämpligt för att lösa dem.

e) Programmeringsspråken C++ och Java associeras oftast med det objektorienterade paradigmet. Detta är eftersom båda språken har bra support för det paradigmet.

f) Ett högnivå-språk är lättare för oss människor att både förstå och översätta till. Att skriva maskinnära skapar risken för att skriva en bug lättare.

   Högnivå-språk är mer portabla än maskinkod. Detta innebär att program skrivna i ett högnivåspråk ofta kan köras på olika typer av datorer utan större eller inga modifieringar.

g) Byggstenen selektion handlar om att programmet ska exekveras från start till slut. Iteration handlar om att programmet ska kunna gå in i en loop och göra samma

exekvering flera gånger om. Selektion handlar om programmet ska göra utföra en viss kod om ett visst kriterium uppfylls.

Uppgift 2:

a) Ett bra program från perspektivet av en användare ska göra programmet så lätt som möjligt att använda. Att kunna gå från start till slut utan problem, det bästa möjliga programmet som kan skapas utifrån en användares perspektiv.

   Ett bug-fritt program kommer göra användare upplevelsen bättre, om användaren påverkas av buggar kan det förstöra arbete, det kan skapa frustation eller liknande.

   Bra program är lätta att hantera, samt lätta att förstå. Om en användare har problem med att förstå eller använda programmet, då är det designat dåligt.

   Programmet ska kunna göra vad det är avsett att göra. Många program fokuserar på att göra sakerna ovan, men glömmer bort att skapa ett program som fungerar. Om programmet inte kan göra sitt jobb, har programmet misslyckats med sitt syfte.

   Ineffektiva program skapar en upplevelse av frustration, användaren vill att programmen ska kunna göra sitt jobb snabbt, utan att behöva vänta på programmet ska ladda klart eller likande. Ett långsamt program kommer skapa en dålig upplevelse.

   Underlättar för handikapp, ett bra program underlättar folk med handikapp. Om användaren av programmet lider av ett handikapp, till exempel är användaren färgblind och programmet har support för färgblindhet så de kan använda programmet. Att inkludera handikapp är lätt att missa men en stor del av populationen lider av handikapp av olika slag. Därför är det viktigt att få tillgång till program som tillåter användare använda program oavsett handikapp.

b) Analys: Bearbetar input från kunden och producenten för att förstå problemet/behovet. Output är en analysrapport.
   Design: Bearbetar analysrapporten för att komma fram till en lösning på problemet/behovet. Output är en designrapport.
   Implementation: Bearbetar designrapporten för att komma fram till koden som ska skrivas. Output är koden.
   Test och integrering: Bearbetar koden för att säkerställa att den fungerar som den ska. Output är testrapporter.
   Underhåll och drift: Bearbetar testrapporterna för att säkerställa att systemet fungerar som det ska. Output är underhållsrapporter.

c) Rapid prototyping är en modell som utgår ifrån tidigare prototyper, där en analys och design gjorts sedan innan. Modellen har sina styrkor i att programmet blir snabbt gjort eftersom grunden finns sedan tidigare. Nackdelen ligger i modellen, den bygger på att ett arbete ska utförts tidigare.

d) Pair programming är ett effektivt sätt för två programmerare att arbeta tillsammans på ett projekt. Den så kallade föraren skriver kod medan observatören granskar varje rad, sedan byter de roller ofta.

Detta hjälper till att säkerställa att koden är felaktig och av hög kvalitet.
Men om personerna som jobbar tillsammans inte är kompatibla med varandra kan arbete ta skada, samt om ena personen inte är lika bra som den andra på arbetet.

e) Att inte planera projektet tillräckligt utförligt kan ofta vara ett misstag. Ofta får detta väldigt stora ringar på vattnet. Om en viktig detalj har missat kommer de i bästa fall kunna gå tillbaka och implementera den missade detaljen, men i värsta fall kommer de behöva börja om från början.

   Vi ser ofta hur program, speciellt spel ofta inte testas tillräckligt utförligt innan de släpps för användning. Då innehåller programvaran ofta buggar eller andra fel. Det här skapar ofta frustration hos användarna, vilket leder till förlust i form av pengar för företaget.

Uppgift 3:

a) Det finns flera skäl till att det är viktigt att arbeta på ett strukturerat sätt vid utveckling av en programvara. För det första gör det enklare att hålla koll på olika delar av koden, och för det andra gör det enklare att återanvända koden. Dessutom gör det enklare att felsöka och uppdatera koden, eftersom det är lättare att hitta felaktig kod i en strukturerad kodbas.

b) Att initieras en variabel betyder att man tilldelar ett värde och en möjligtvis datatyp. Medan att deklarerar betyder att man skapar en variabel utan ett värde.

c) Koka pasta. Man. Börja med att kolla om vatten finns, om pasta finns, om en kastrull finns och en spis eller likande finns. Sedan går man vidare med att fylla upp en kastrull med vatten, efter det låter man vattnet koka upp på värmekällan. När vattnet kokat upp häller man ner pastan och efter minuterna som finns på pasta paketet häller man av pastavattnet. Klart.

d) Linjärsökning är en sökalgoritm som går igenom en lista av element från vänster till höger eller från höger till vänster och returnerar det element som matchar sökkriteriet. Exemplet nedan går igenom en lista av namn från vänster till höger och returnerar det första namnet som matchar sökkriteriet:
Namnlista: ["Adam", "Bertil", "Ceasar", "David", "Erik"]
Sökkriterium: "E"
Returnerar: "Erik"

Binärsökning är en sökalgoritm som går igenom en sorterad lista av element. Algoritmen börjar från mitten och sedan söker antingen vänster eller höger beroende på om värdet är större eller mindre, den här sökningen upprepas tills värdet hittas eller inte hittats.

Exempel: Lista: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Element som söks: 5

Jämför 5 med 1: 5 > 1, gå höger

Jämför 5 med 3: 5 < 3, gå vänster

Jämför 5 med 2: 5 > 2, gå höger

Jämför 5 med 4: 5 = 4, elementet hittades

e) Bubbelsortering går igenom en array och jämför varje värde med det värdet som kommer efter det. Om det första värdet är större än det andra värdet, byter de plats. Om det första värdet är mindre än det andra värdet, byter de inte plats. Detta görs för varje par av värden i arrayen. När det gått igenom hela arrayen en gång, kommer det minsta värdet att vara först i arrayen. Detta görs om och om igen tills arrayen är

sorterad.

I arrayen [1, 4, 2, 3] kommer det betyda att den jämför värderna 1 och 4, värderna kommer inte byta plats. Värdarna 4 och 2 jämförs sedan och byter plats därefter. 4 och 3 jämförs och sedan byter plats. Arrayen är slut, nu kommer den börja om igen och jämföra alla element i arrayen igen. Detta kommer ske tills den är sorterad.

Steg 1: [1, 4, 2, 3]
Steg 2: [1, 2, 4, 3]
Steg 3: [1, 2, 3, 4]
Steg 4: [1, 2, 3, 4]

f) Fördelarna med att dela upp programkod i funktioner är många. Funktioner gör det möjligt för oss att skapa mindre och mer hanterbara bitar av kod, och det gör det också möjligt för oss att återanvända koden.
Funktioner gör det också enklare att debugga och testa koden, eftersom vi kan testa varje funktion för sig.

g) 1. Kompilatorer: Kompilatorer är verktyg som omvandlar källkod till binärkod. De flesta kompilatorer har optimeringsfunktioner som gör att koden körs snabbare.

2. Debuggers: Debuggers är verktyg som hjälper till att hitta och åtgärda fel i koden. De flesta debuggers har breakpoints-funktioner som gör att man kan ställa in punkter där koden ska stoppas och man kan sedan undersöka variabler och andra data.

3. Dokumentation: Dokumentation är ett verktyg som hjälper oss förstå koden bättre. Dokumentation innehåller vanligtvis information om kodens syfte, hur den fungerar och exempel på hur den används.

Uppgift 4:

a) Data kan struktureras på ett ändamålsenligt sätt genom att använda arrayer, strukturer och databaser. Exempel på detta är att använda en array för att lagra en lista med namn, en struktur för att lagra information om en person eller en databas för att lagra information om olika produkter.

b) Fördelarna med att strukturera data på ett ändamålsenligt sätt är många. För det första gör det det enklare för människor att hitta och använda data. För det andra gör det det enklare för dataprogrammerare att hantera data. För det tredje minskar det risken för felaktigheter i data. För det fjärde gör det det enklare att överföra data mellan olika system. För det femte gör det det enklare att uppdatera och ändra data.

c) Datastrukturen array är en lista med element av samma datatyp, datan är sparad sekventiellt i minnet. Elementen börjar från index 0. Element kan ses som skåp på ett led. Varje element har ett index (egen adress). Kan kallas ibland (endimensionell) vektor.

   En struct är en datastruktur där grupper av datatyper kan sparas. Skillnaden mellan en array och en struct är att structen kan spara till exempel både en integer och en string, medan en array enbart skulle kunna spara en datatyp åt gången.

   Stack datastrukturen är byggd på konceptet sist in, först ut (LIFO). Varje gång ett element läggs till i en stack placeras den på högst upp på 'högen'. När man ska ta bort ett element tas det senast tillagda elementet bort först.

   Datastrukturen kö fungerar likt det mänskliga kö konceptet. Där gäller det att nya element läggs sist kön och gamla tas bort först i kön. Med andra ord ligger det äldsta elementet överst, det nyaste ligger underst.

d) Rad 1: Initierar en variabel av typen int med namnet tal1 och tilldelar den värdet 5.
   Rad 2: Initierar en variabel av typen int med namnet tal2 och tilldelar den värdet 3.
   Rad 3: Initierar en variabel av typen int-pekare med namnet pekare och tilldelar den adressen till tal2.
   Rad 4: Subtraherar 1 från tal1.
   Rad 5: Subtraherar 1 från tal2.
   Rad 6: Dividerar tal1 med tal2 och tilldelar resultatet till tal2.
   Rad 7: Pekare tilldelas värdet av pekare och adderas med sig själv.
   Rad 8: Skriver ut tal1 på skärmen.
   Rad 9: Skriver ut tal2 på skärmen.
   Rad 10: Skriver ut värdet av pekare på skärmen.

e) Fel 1, rad 1: #include <sodastream>, det är inte ett bibliotek som finns med i standard c++, det kan däremot vara en egengjord header-fil. Däremot i detta program är det <iostream> som önskas.

   Fel 2, rad 11: array[2]=tal2, arrayen ogiltig, eftersom tidigare deklarerades arrayen

att vara 2 element lång. Den försöker tilldela element 3 ett värde, men element 3 är ogiltigt. Den hade blivit giltig om array[1]=tal2.

Fel 3, rad 13: Funktionen main() saknar en slut måsvinge.

Fel 4, rad 12: raden avslutas med en parentes. För att lösa problemet behöver den tas bort. Kan lösas på andra sätt, men detta är snabbaste och enklaste lösningen.

Fel 5, rad 5: saknar semikolon efter temp=0.

Fel 6, rad 12: koden behöver skrivas om från ”cout<<array[0]<<end<<array[1];)” till ”cout<<array[0]<<endl<<array[1];” om den ska vara giltig.

Den bugfria koden hade sett ut:

```
#include <iostream>
using namespace std;

int main(){
 int tal1=11, tal2=22, temp=0;
 int array[2];
 temp=tal1;
 tal1=tal2;
 tal2=temp;
 array[0]=tal1;
 array[1]=tal2;
 cout<<array[0]<<endl<<array[1];
}
```