Hemtentamen

Grundläggande programmering med C++

Uppgift 1

A. Ett programmeringsparadigm är ett tankesätt eller angreppsätt om hur man programmerar ett program.

B. Procedurell programmering är set som ett HUR perspektiv på programmering.
Procedurell programmering är baserat på kontrollstrukturerna Sekvens, Iteration, Selektion och Funktion.
Det Procedurell programmering säjer att man ska dela upp program i flera delprogram (oftast i funktioner) som ska föröka vara så självständiga som möjligt.
Detta görs då enligt Procedurell programmering ska det vara enklare att hantera komplexitet och att hitta problem/buggar i koden om alla delar av ett program är uppdelade.
Procedurell programmering går också ut på att om något går fel i en av delprogrammen ska fortfarande huvud programmet kunna köras utan att krascha.
Några fördelar med Procedurell programmering är att det är enkel att förstå och felsöka då allt är uppdelat i delprogram.
En svaghet med Procedurell programmering är om man till exempel skulle ha två spelare till ett spel så skulle man behöva göra minst två delprogram (troligt vis fler) för att kunna hantera spelarna så det kan snabbt bli mycket kod som får repeteras om man vill göra större program.

C. Båda paradigmen utgår ifrån ett VAD perspektiv men det objektorienterade programmering har är att tänka på alla program som objekt som ska innehålla både operatorer och enheter.
En styrka med objektorienterade programmering är objekten med att man kan skapa ett objekt till till exempel bilar så behöver man inte skapa samma kåd till flera bilar utan kan bara använda objektet själv till det.
En svaghet med objektorienterade programmering är att det kan vara svårt att förstå om man inte kan programmering sän innan så det är inte nybörjarvänligt.

D. Ett problem där det deklarativa paradigmet skulle passa är om man skulle hämta data ifrån en databas då fokuset på VAD det är för data göt det enkelt att få tag på den data man vill ha.

E. C++ och java

F. Högnivåspråk har fördelarna att det är väldigt mycket enklare för en människa att förstå och läsa med också att det inte är platforms beroende så det kan fungera på olika typer av datorer.

G. Selektion (if sats) är ett val i koden där programmet måste välja en väg att ta. Det kan finnas flera vägar men det måste alltid finnas minst två vägar i en selection.
Iteration (loop) är en upprepning av programmet medans ett villkor är sant. En Iteration kan köras 0 eller flera gånger men ska helst ha ett slut någon gång.
Sekvens är att program ska ha en början och ett slut som är vart programmet ska börja och vart det ska sluta.

Uppgift 2

A. Utifrån ett användares perspektiv ska ett bra program vara billigt (om det kostar pengar) då användare vill spendera så lite pengar som möjligt, enkelt att använda då användare inte brukar gilla att behöva läsa igenom manualer för att använda program, snyggt (utifrån grafik) då användare brukar föredra program som inte ser hemska ut, bugg frit då användare tycker det är jobbing om något inte vill funkar som det ska, inte tar upp för mycket minne/RAM då användare brukar villa kunna ha fler än ett program på sin dator samtidigt.

B. Första fasen av Livscykelsmodellen är analys och är där man får ett problem som ska lösas och ska sedan försöka hitta antaganden och krav till lösningen och det man ska få av analysen är en kravspecifikation.
Andra delen av Livscykelsmodellen är design vilket är där man ska ifrån kravspecifikationerna skapa en design på hur programmet ska byggas vilket kan inkludera delproblem, vilket platform programmet ska göras på men även vilket språk som ska användas. Det man ska få av design är en System beskrivning.
Nästa del av Livscykelsmodellen är implementation vilket är att faktisk skapa programmet ifrån designen man har. I implementation ingår även att testa specifika delar av koden. Det man ska få av implementation är version 1 av programmet.
Delen efter implementation är testing och installation viket är där man går igenom hela programmet man har för att försöka hitta buggar och testar programmet under olika situation för att se om det kan hantera dom. Det man ska få ifrån implementation är ett fungerande program.
Sista delen av Livscykelsmodellen är underhåll och drift vilket är att hålla programmet fungerande (till exempel hålla servrar uppe) och uppdatera det om det behövs.

C. Rapid prototyping går utt på att man ska skapa många prototyper av ett program för att försöka hitta problem med designen/iden av programmet.
En styrka med rapid prototyping är att man kan snabbt hitta problem med designen/iden och försöka fixa dom innan det gått för långt. En svaghet med Rapid prototyping är att det kan ta ganska lång tid att komma igenom prototyp fasen så man kan behöva spendera mycket tid och pengar där.

D. Pair programming (också kallad Extreme programming) går utt på att ha en focus på kunden och att vara snabb. Pair programming fungerar också genom att det ska vara ett par som programmerar så en person ska sita vid datorn och programmera medans en annan ska sita bredvid och se till att programmeraren inte gör fell och följer designen.
En styrka med pair programming är att man kan snabbt få program gjorda och fel brukar kuna upptäckas snabbt då man är två som kollar på koden. En svaghet med pair programming är att det ofta blir fel/buggar så man behöver ofta göra om saker men också att programmerarna måste kunna arbeta bra tillsammans annars blir det strul.

E. Ett vanligt misstag är att man direkt börjar med implementation av programmet utan en design. Det brukar ledda till att man kommer behöva göra om mycket av koden då det ofta kan bli stora problem om man inte har en design.
Ett annat problem är att man glömmer eller inte tänker på dolda krav som program har. Det kan ofta leda till att man missar viktiga delar eller gör antaganden som kunden inte gillar så man får börja om eller göra om mycket av koden.

Uppgift 3

A. Att arbeta på ett strukturerat sätt i programmering kan bero lite på men några vanliga sät är att dela upp sin kod i delar så det blir enklare att hålla koll vad varje del av programmet gör. Ett annat sätt är att skriva kommentarer i koden för att kunna komma ihåg vad kod gör men också så om någon annan person skulle få

ens kod kan dom enklare förstå en kod. Andra sät är att man indenterar kod vilket gör att det blir enklare att se om koden tillhör en funktion eller loop. Anledningen man ska ha bra strukturerad kod är för att det blir mycket enklare för en själv men även andra att arbeta och förstå koden.

B. Att deklarera en variabel är att man bara skapar variabeln (till exempel int number) medans att initiera en variabel är att man skapar den och tilldelar den ett värde samtidigt (till exempel int number = 4)

C. sätt sig i bilen
sätt i nycken
vrid om nycken
starta motorn
tryck på gas pedalen

D. Linjärsökning är att programmet ska söka igenom en array ifrån början till slutet tills den hittar alla svar. Linjärsökningar är enkla att implementera och förstå men kan ta lång tid om det blir långa arrayer eller komplicerade jämförelser.
Binärsökning är att programmet börjar i mitten av arrayen och kollar om det den söker efter är mer, mindre eller samma som mitten och kan sen ta bort hälften av arrayen om den inte hitta svaret och börja om i mitten av det som är kvar tills den har hitat svaret. Binärsökning är mycket snabbare än linjärsökningar och mer effektiv men kräver att arrayen är sorterad och kan vara svårare att förstå.

E. Bubbelsortering (kommer användas som minst först) fungerar genom att den går igenom varje tal i arrayen och kollar om talet till höger är mindre eller större och är det mindre så byter den plats på dom. Bubbelsorteringen kommer fortsätta så tills kan gå igenom hela arrayen utan att behöva byta några tal.

Så när bubbelsorteringen ska gå igenom [1 , 4 , 2 , 3] kommer den börja med att kolla om 4 är mindre än 1 vilket det inte är så inget händer men när bubbelsorteringen kollar nästa position om 2 är mindre än 4 så kommer den byta plats på dom så arrayen blir [1 , 2 , 4 , 3]. Efter att ha byt plats på 2 och 4 kommer bubbelsorteringen fortsätta med om 3 är mindre än 4 och byta på dom så arrayen blir [1 , 2 , 3 , 4]. Nu när bubbelsorteringen har nått slutet av arrayen så börjar den om ifrån början och går igenom arrayen igen för att se om något behöver flyttas men eftersom det inte är något kvar att flyta på så kommer den att nå slutet av arrayen utan att flyttat på något och avsluta bubbelsorteringen där.

F. Att dela upp programmet i funktioner hjälper då det blir enklare att se vad som händer i programmet om allt inte är i main. En annan fördel är att det blir enklare att debugga om programmet är uppdelat då man kan se vilken funktion det är som problem händer i. Funktioner hjälper också om man ska göra samma sak fler gånger då man inte behöver skriva samma kod flera gånger utan kan istället använda en funktion för att hålla koden och sen bara använda funktionen. Funktioner kan också vara bra för säkerhet då man kan gömma händelser lättare i funktion så det inte går att komma åt dom lika lät utifrån.

G. Ett hjälpmedel är auto indentering då programmet blir enklare att förstå om saker är indenterade så att kunna automatiskt indentera hjälper. Bibliotek är också en väldigt hjälpsam sak då programmeraren sliper behöva till exempel skapa sin egen cout utan kan istället importera en genom ett bibliotek. Ett annat hjälpmedel är kodning program som till exempel visual studio code då dom hjälper att hålla koll på ens kod men också att stava korrekt men även skapa delar av ens kod.

Uppgift 4

A. Så man kan strukturera data på i ett program beror lite på vad det är man ska strukturera men har man till exempel en variabel pengar som håller koll på hur mycket pengar en spelare har så räcker det med att ha den deklarerad i början av programmet. Ska man dock ha till exempel något som ska hålla koll på alla spelarens insatser så kan man använda en array för att hålla alla data i då det blir enklare om alla data är på samma ställe och arrayer kan skapas utan en max gräns så man inte behöver vara räd att få slut på plats. Skulle man behöva en lista med olika typer av element i så kan man skapa en struct som är lik en array fast kan hålla olika typer element dock så tar den mer minne en en array. Om man ska ha en lista som saker ofta kommer ändar sig i kan man använda en link list viket kan vara svårt att sätta up men den tillåter en att enkelt ta bort eller lägga till saker i listan och är enkla på minnet.

B. Anledningen man skulle strukturera data på ett ändamålsenligt sätt är för att det gör hanteringen av data mycket enklare. Att strukturera data bra gör att programmet blir enklare att förstå då data inte behöver flytas/bytas hela tiden och det hjälper minnes hanteringen då det blir lättare för datorn att hålla all data. Bra datastruktur kan även göra att programmet går snabbare då programmet inte behöver gå igenom lika mycket för att hitta data det behöver.

C. Arrayer och structures är ganska lika varandra med att dom båda strukturerar samman data som hör ihop. Enda skillnaden är att i Arrayer har all data samma datatyp medans i structures är kan data vara olika datatyper. Stack och kö är också ganska lika med varandra med att båda handlar om hur element hamnar i listor men skiljer sej åt vart elementen ska hamna i listor. Stack använder principen av LIFO (sist in, först ut) alltså senaste elementet som senast hamnade i listan är det som åker ut först. Kö använder principen av FIFO (först in, först ut) alltså första elementet i lista är det som åker ut först.

D. Det som händer i detta program är att i början så sätts tal1 till 5, tal2 till 3 och *pekare till 3 också då *pekare pekar på tal2.
Efter att talen sätts så minskas tal1 och tal2 med 1 vilket också gör att *pekare minskar med 1 så talen blir tal1 = 4, tal2 = 2 och *pekare = 2.
Nästa del är uträkningen av tal2 = (tal1-tal2)/tal2 vilket om man säter in vad talen är blir tal2 = (4-2)/2 viket blir 1 så tal2 = 1 viket gör att *pekare också blir 1 då den fortfarande pekar på tal2 så talen är nu tal1 = 4, tal2 = 1, *pekare = 1.
Sista uträkningen är *pekare = *pekare+*pekare vilket om man säter in talen blir *pekare = 1+1 så *pekare = 2 men eftersom *pekare pekar på tal2 så byte tal2 också till 2 så det som skrivs ut efter det i couten blir 4 2 2.

E. Några fel som jag kan hitta i koden är

- det finns inget bibliotek som heter sodastream.
- main stängs aldrig då det är fel parentes på slutet.
- koden *int tal1=11, tal2=22, temp=0* har inget semikolon på slutet.
- platsen array[2] finns inte då array bara är 2 stor.
- i cout så har någon glömt l i endl.
- array[0] sätts aldrig så att försök skriva ut den ger slumpmässiga tal.

så här skulle jag fixa programmet

```
#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];
}
```