Problemlösning

En viktig kunskap som alla utvecklare måste kunna är problemlösning. Problemlösning förknippas oftast med matematik men är i grund och botten en konst att få ett större problem att bli flera mindre lösbara problem.

Robot problemet

Tänk er att vi har en robot som just nu är livlös och bara står still. Vi vill lära denna robot att hämta ett glas vatten från köket. När vi nu säger åt vår robot att hämta vårt glas med vatten står den helt livlös och förstår inte vårt kommando. Detta beror på att roboten förstår inte vad vi menar med vårt kommando då roboten inte har någon aning om var “gå”, “hämta”, “glas” och “vatten” betyder och det är vårt jobb att lära roboten vad den ska göra. I helheten är problemet att roboten ska kunna hämta ett glas med vatten till oss, och detta problem ska vi bryta ner till mindre problem.

Vi börjar med att lära roboten att gå. För sakens skull så förenklar vi detta då det skulle kräva för mycket text att definiera allt. Robotens nuvarande läge är stillastående med båda benen raka bredvid sig. Vi instruerar roboten att när roboten är i detta läge så ska han röra ena benet framåt en halvmeter, vi väljer höger ben. Nu är roboten i ett helt annat läge, det vill säga att höger ben är längre fram än vänster, så vi instruerar roboten att röra vänster ben framåt så att benet hamnar en halvmeter före höger ben. Nu har vi än en gång ett nytt läge, men detta läge är samma som förra ben så vi instruerar roboten att röra höger ben igen. Dessa instruktioner för benen kallar vi nu för “gå” och sparar informationen i robotens minne.

Nu bör vi definiera “vatten” och “glas” för roboten. Vi beskriver utförligt vad ett glas är och vad det består av och samma sak med vatten. Nu måste vi däremot ge instruktioner vad dessa två kommandon betyder när de förekommer i samma kommando. Vi förklarar att roboten ska med

ena armen och handen greppa tag i ett glas och sedan fylla glaset med vatten tills vattnet nästan nått glasets topp. När detta har uppfyllts så håller roboten i ett “glas med vatten” vilken ska returneras till den som gav kommandot.

Nu ska vi se till att roboten uppfyller kommandot “hämta”. Här förklarar vi för roboten att om robotens position inte är i närheten av både “vatten” och “glas” så bör roboten använda sig av funktionen “gå” för att ta sig till båda objekten i den ordning som “glas med vatten” kräver. Därefter ska vi instruera att “hämta” betyder att den ska först uppfylla kravet “glas med vatten” genom att använda sig av “gå”, därefter ska roboten igen använda sig av “gå” för att ta sig till personen som gav kommandot och därefter ge den personen objektet genom att räcka ut armen med objektet. Detta uppfyller kommandot “hämta”.

Nu har vi (väldigt simplifierat) lärt en robot att hämta ett glas med vatten genom att bryta ner hela problemet till flera små problem som vi enkelt kan lösa enskilt.

Flödesschema

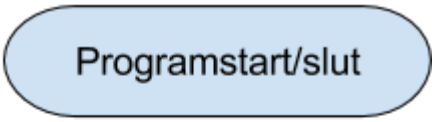
Flödesscheman används för att skapa en visuell bild över den logik vi ska implementera och dessa skapar vi innan vi sätter oss och börjar koda. Med hjälp av flödesscheman effektiviserar vi vårt arbete markant i och med att vi får en god överblick av vad vi ska skapa samt att vi näst intill skapar en steg-för-steg ritning för hur vi ska implementera vår kod.

Tips: Penna och papper är din bästa vän när det gäller snabb dokumentation. Det snabbaste sättet att skapa ett flödesschema är med penna och papper men det finns program/tjänster där du kan skapa ditt flödesschema.

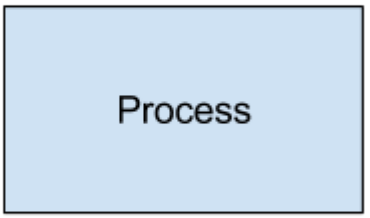
Så hur skapar vi ett flödesschema och hur ser det ut?

När vi ritar upp vårt flödesschema så använder vi oss av olika grafiska former för att indikera programstart, process, input/output och så vidare och dessa kopplas ihop med pilar som visar flödet i programmet.

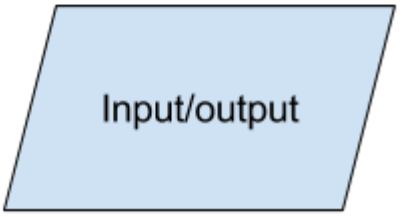
Denna symbol används för att indikera var programmet börjar och slutar



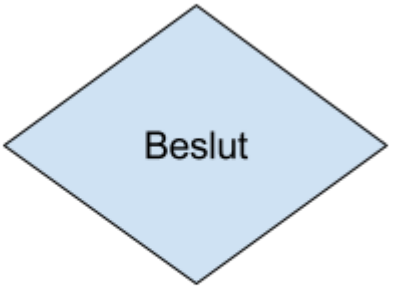
Rektangeln indikerar att programmet genomgår en process. Exempelvis adderar två tal, loopar 10 gånger och så vidare



Denna symbol används för att indikera att vi kräver ett input från användaren. Den används även till output, där vi exempelvis skriver ut text till en konsol



Denna symbol används vid beslut där resultatet kan vi true eller false



Pilar används för att länka olika former och skapa programmets flöde

Exempel på ett program som tar in ett tal mellan 1-10 och ger tillbaka talet i kvadrat: