

INNEHÅLLSFÖRTECKNING

SYFTE.....	2
SYSTEMKRAV OCH UPPHOVSRÄTT.....	2
STARTA FLOPP ÄR ENKELT... ..	2
...MEN MAN MÅSTE VETA VAD MAN SKA GÖRA FÖRST!	3
ETT ENKELT PROGRAM KONSTRUERAT I FLOPP (MODULTYP: START OCH END).....	3
MODULTYP: TILLDELNING (ASSIGNMENT) AV VARIABEL OCH ARITMETISKA UTTRYCK	3
MODULTYP: IN OCH UTMATNING (INPUT & OUTPUT)	3
MELLANSPEL - KONSTRUERA OCH KÖRA ETT LITET PROGRAM	4
<i>AddBox</i>	4
<i>Flow</i>	4
<i>MoveBox</i>	4
<i>Delete</i>	5
<i>EditBox</i>	5
<i>Spara och ladda en konstruktion</i>	5
<i>Exekvera ett program</i>	5
MODULTYP: SELEKTION (IF)	6
MODULTYP: PROCEDURER (PROC, DEFPROC & RETURN)	6

Syfte

Flopp står för "Faked Language On a Programmable Pattern board"

Programmet Flopp är tänkt att användas innan man lärt sig ett formellt språk och är tänkt att illustrera

- Enkla beräkningar med tilldelningar av variabler
- Selektion {TRUE, FALSE}
- Funktionsanrop

Som namnet antyder använder Flopp inte något riktigt språk utan ska endast användas för att i början av en programmeringsteknikkurs illustrera hur en dator arbetar i grunden. Det har varit programkonstruktörens intention att Flopp ska vara enkelt, intuitivt och användas i problemlösningssammanhang eller för att testa idéer och algoritmer. Om någon tycker att programmet inte lever upp till detta eller är lämpligare att användas i andra sammanhang ber han att vederbörande hör av sig till honom.

Systemkrav och upphovsrätt

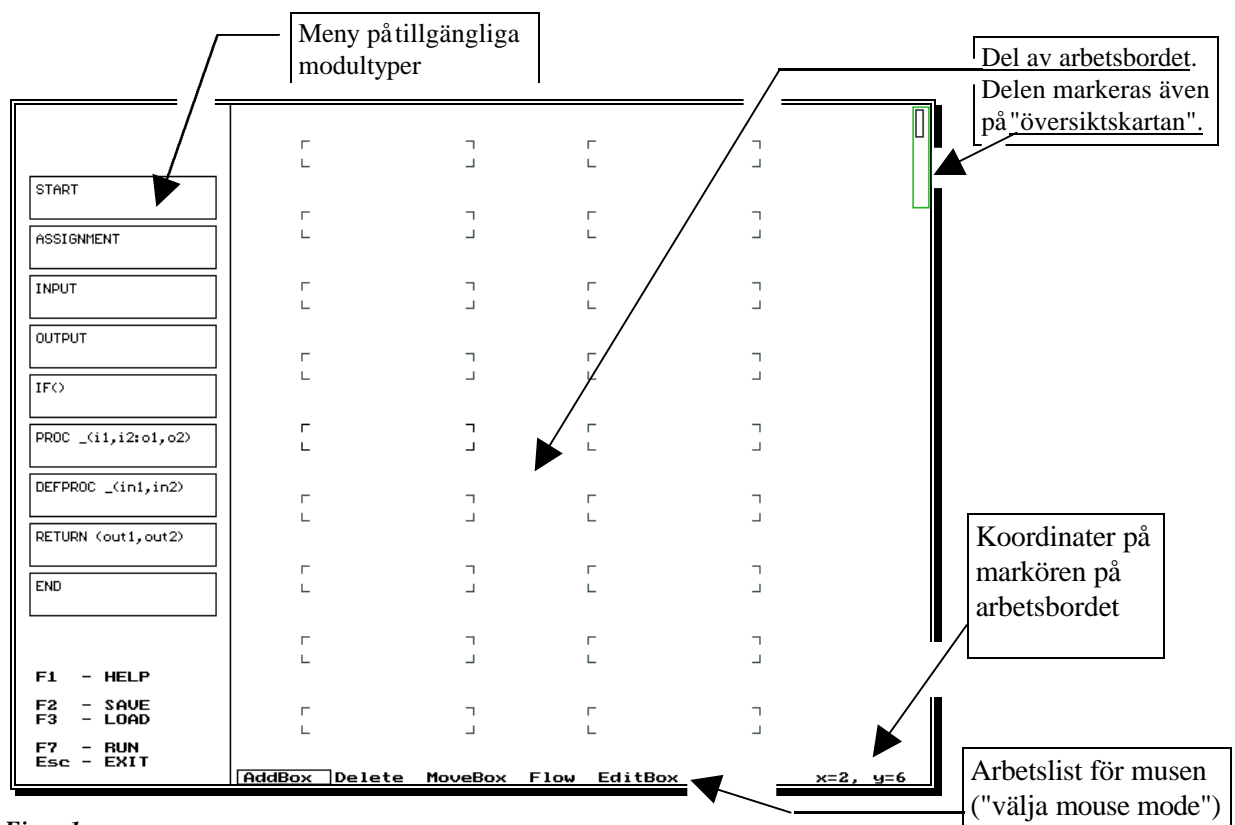
Programmet kräver

- MS-DOS operativsystem (dvs. gå i DOSfönster under Windows 95/NT),
- minst VGaskärm som klarar 640x480 pixlars högupplösningsgrafik
- mus.

Programvaran får ej användas av andra än studenter och anställda vid Mälardalens högskola. Programvaran får därför inte säljas, kopieras, hyras ut eller på annat sätt spridas eller mångfaldigas utanför Mälardalens högskola. I alla andra fall gäller lagen om upphovsrätt (vilket kan vid rättsprövning i Sverige medföra böter eller fängelse i högst tvåår).

Starta Flopp är enkelt...

Programmet startas igång antingen direkt ifrån DOS genom att skriva "FLOPP" vid prompten eller med ett (dubbel)klick på Flopp-ikonen i Windows. Skärmen ska då se ut enligt (Figur 1) och programmet är färdigt att använda. Du kan nu antingen ladda in en fil genom att trycka F3 eller börja göra en egen konstruktion. Programmet går att använda helt med tangentbordet, men de flesta funktioner understöds även av musen.



Figur 1

...men man måste veta vad man ska göra först!

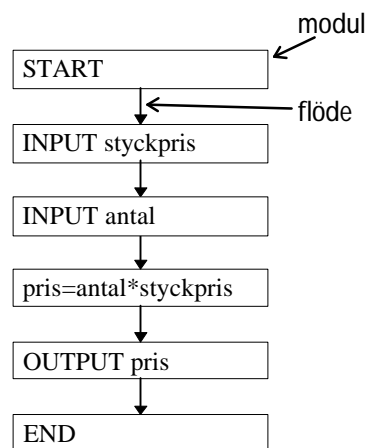
Att klicka hejvilt och hacka ihop lite kod kan nästan vem som helst. Efter sju sorger och åtta bedrävelser så kan alstret till och med fungera som det var tänkt. Denna metod (även kallad "Big-bang-hack") är varken särskilt effektiv eller ingenjörsmässig och det finns en betydligt bättre väg. Det tar visserligen en stund att lära sig metoden men denna tid får man tusenfalt (utan överdrift!) igen eftersom programmen blir rätt från början. Metoden kan sammanfattas i en enda mening ; "Tänk igenom först **vad** du ska göra innan du ens funderar på **hur** du ska lösa det." I alla andra sammanhang måste man ju göra så och programmeringssammanhang är inget undantag. Visserligen går ingenting sönder genom det vilda klickandet, men man sparar mycket tid och frustration genom att tänka först genom att kanske skissa lite med papper och penna på en lösning.

Ett enkelt program konstruerat i Flopp (Modultyp: Start och End)

I Flopp kan program både konstrueras och köras och de kommer att se ut som enklare *flödesdiagram*. Programmen byggs upp av *moduler* som bara kan en sak t.ex. beräkna ett uttryck, tilldela en variabel, jämföra två tal eller uttryck, anropa en process eller skriva ut något på skärmen. Det går således inte att både göra en utskrift och en jämförelse i en modul. För att göra något meningsfullt så måste således flera moduler knytas ihop till ett komplett program.

När programmet är färdigkonstruerat så kan du prova om det fungerar eller t.o.m. använda det "på riktigt" och det kallas att man "kör" eller *exekverar* programmet. Flopp kan endast exekvera programmet steg för steg (dvs. det går inte att "starta" programmet och låta det köra av sig självt). Trådarna som binder ihop modulerna kallas *flöden* eller flödespilar och är enkelriktade. Med det menas att modulerna måste köras i en viss ordning - en fix *sekvens*. Program i Flopp måste alltid börja med modulen START och avslutas (*termineras*) med modulen END.

I Figur 2 illustreras en enkel sekvens av moduler som tillsammans utgör ett program som beräknar ett totalpris på ett variabelt antal varor med ett visst styckepris.



Figur 2

Modultyp: Tilldelning (assignment) av variabel och aritmetiska uttryck

Tilldelningsmodulen är till för att tilldela en variabel ett värde. (I Flopp behöver variabler inte deklarerars utan det görs de automatiskt första gången de tilldelas). Variabelnamnen kan se ut hur som helst så länge de bara innehåller bokstäver och även siffror om variabelnamnet inleds med en bokstav. Observera att Flopp skiljer på versala och gemena (dvs. namn \neq naMn). Om en variabel med ett namn redan tilldelats i en tidigare modul, så kommer denna att skrivas över med det nya tilldelade värdet vid en ny tilldelning av variabelnamnet. Tilldelning sker alltid från höger till vänster. Högerledet kan vara ett *uttryck (expression)* som innehåller både konstanter och variabler som tilldelats värden i tidigare moduler *.

Flopp klarar de fyra räknesätten {+, -, *, / (heltalsdivision)} samt parenteser { (,) } och modulus { % }. Parenteser har högst prioritet och subtraktion och addition lägst. Om en odefinierad variabel förekommer i uttrycket kommer beräkningen gå fel och därmed även tilldelningen. Observera att ordet "assignment" inte skall ingå i en redigerad modul då den utgör bara en ledtext om modulen är oredigerad.

Modultyp: In och utmatning (input & output)

Med modultyperna INPUT och OUTPUT kan programmet interagera med användaren (personen bakom tangentbordet). Programmet begär att användaren ska mata in ett värde i en INPUT-modul som kommer tilldelas en viss variabel. Variabler skapas och tilldelas på samma sätt som vid tilldelning.

Med OUTPUT kan programmet skriva ut ett resultat på ett litet fönster i Flopp. "Resultatet" kan vara ett godtyckligt uttryck. På nästkommande sida följer tre exempel:

* Det är således omöjligt att skriva "7*6=var". Det är även omöjligt att göra $x+y=9-2$ eftersom =-tecknet betyder tilldelning av en variabel och är således ingen ekvation.

- OUTPUT bredd
- OUTPUT bredd+7
- OUTPUT 7

Det första exemplet visar utskrift av en variabel, det andra ett sammansatt med en variabel och konstant uttryck och i det sista skrivs en konstant ut. Om en odefinierad variabel används i OUTPUT genereras en felutskrift eller så blir utskriften fel (det är skillnad på de bägge!). Endast en variabel kan tilldelas i INPUT och endast ett uttryck kan skrivas ut på skärmen med OUTPUT.

Mellanspel - konstruera och köra ett litet program

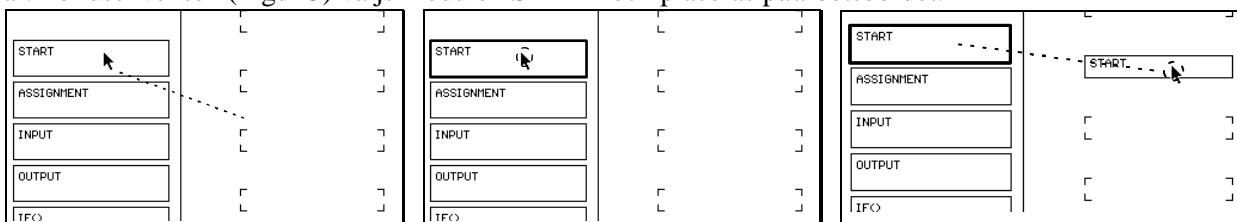
Innan resten av modulerna förklaras så följer här ett litet exempel på hur ett program skapas från början i Flopp och mellanspelet avslutas med en programkörning. Programmet som ska skapas är det lilla som beskrevs i samband med tilldelning. Denna beskrivning utgår från att konstruktören har kommit fram till att detta program är en lösning. Det stora "huvudarbetet" med problembeskrivning, kravspecifikation etc. anses i detta skede vara klart och går direkt på hur ett program skrivs in (*implementeras*) i Flopp.

Det kanske är på sin plats att tala om vilka musoperationer man *inte* kan göra:

- Inga objekt eller knappar går att dubbelklicka på - Ett sådant kommer uppfattas som tvåklick.
- Drag-and-drop konceptet är heller inte tillgängligt - dvs. klicka på ett objekt, dra det och släpp det genom att släppa knappen.

AddBox

Börja med att lägga ut de moduler som anses behövas i programmet. Det görs genom att ligga i musläge "AddBox". Klicka på texten AddBox. Genom att välja i modulmenyn kommer en modul att markeras. Denna modultyp kommer nu att läggas i rutnätet om du klickar på arbetsytan. Du kan placera modulen var du vill, men av strategiska skäl så bör den placeras på arbetsytans övre hälft. Du kan flytta dig omkring på arbetsytan med tangentbordets piltangenter. Observera gärna Koordinatangivelserna och översiktskartan vid förflyttningar. I bildsekvensen (Figur 3) väljs modulen START och placeras på arbetsbordet.



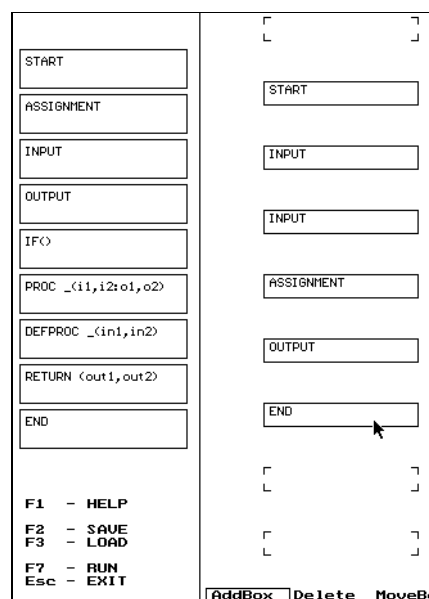
Figur 3 - Streckad linje symboliserar markörförflyttning och ring runt markören markerar vänsterknappsklick.

Flow

Fortsätt att placera ut modulerna på arbetsbordet så att de hamnar enligt (Figur 4) och välj sedan musläge "Flow" på arbetslistan. I detta läge markeras sekvensordningen eller instruktionsflödet som det även kallas. I flowläget väljer du med första klicket var pilen skall börja och med andra var det ska sluta och om din destinationmodul inte är synlig på skärmen så förflyttar du dig dit med piltangenterna. I händelse att du ångrar dig klickar du med höger musknapp eller utanför arbetsbordet.

MoveBox

Om en modul behövs flyttas på arbetsbordet välj "MoveBox" i arbetslistan och klicka på modulen ska flyttas. Modulen kommer nu att försvinna (eller mera exakt - flyttas till en urklippshanterare). Vid nästa musklick kommer modulen med alla flödespilar till och från att klistras in på vald plats.



Figur 4

Delete

Om du vill radera ett objekt så väljer du "delete" i mus-arbetslistan. Klicka på modulen och menyn på tillgängliga datatyper ersätts med en lista på de objekt som kan raderas. Observera att om en flödespil ska raderas så väljs den modul som pilen kommer ifrån.

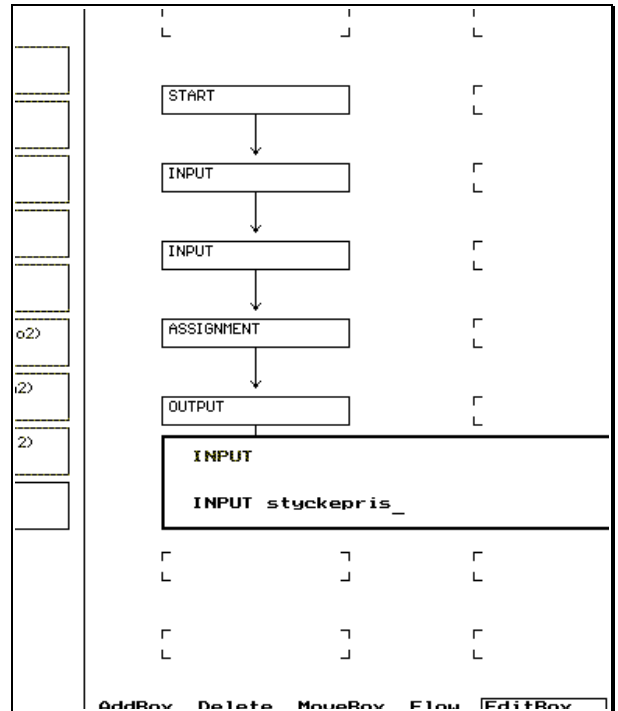
EditBox

När flödespilarna är dragna så måste modulerna få en exakt betydelse eller instruktion då modulen i sig inte säger exakt vad som ska göras. Modulerna i sitt ursprungliga läge talar bara om vilken typ av instruktion den kan innehålla. Själva instruktionen måste specificeras och det görs i listläget EditBox. Då en modul markeras kommer dess textinnehåll eller instruktion att kunna ändras i ett fönster - se Figur 5. Den övre (gula) texten går ej att ändra då den endast beskriver modultypen. Den nedre (röda) texten är däremot instruktionen som ska utföras och kan (ska) därför ändras. I figuren intill är det en INPUT-modul som ska redigeras och då instruktionen INPUT bara talar om att "något" ska matas in och tilldelas så måste vi lägga till var detta något ska tilldelas. Vi ska med andra ord specificera ett variabelnamn - i detta fall "styckpris". Kompletteringen avslutas med Returtangenten.

Om man av någon anledning ångrar sig och inte vill att den redigerade instruktionen ska läggas in i modulen så används Esc-tangenten.

Notering 1: START- och END-modulen redigeras aldrig.

Notering 2: Även om inte all text syns i "modulboxen" så "finns" den där.



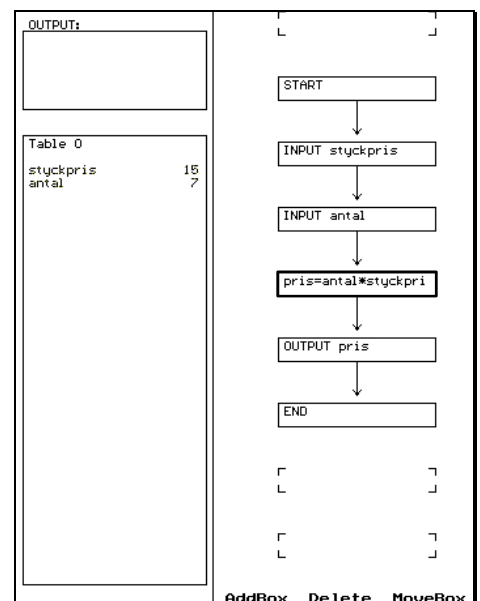
Figur 5

Spara och ladda en konstruktion

I Flopp kan användaren närhelst spara (F2) sina alster och ladda (F3) gamla. Då man trycker någon av dessa knappar kommer ett fönster upp som frågar vad konstruktionen som ska laddas/sparas ska ha för filnamn. Till vänster om fönstret listas de filer som befinner sig i den aktuella katalogen (contents). Flopp varnar när en redan befintlig fil med identiskt namn riskeras att skrivas över.

Exekvera ett program

I Flopp kan man köra sina program steg för steg (egentligen modul för modul). Genom att trycka på F7 kommer den markerade modulen att exekveras och därefter gå till nästa modul. Observera att den markerade modulen inte har exekverats förrän F7 har tryckts. Om ingen ruta är markerad (t.ex. direkt efter en redigering av programmet) kommer Flopp automatiskt att söka sig till STARTmodulen. Variabler som används eller definieras kommer automatiskt att visas till vänster om arbetsbordet i ett särskilt fönster kallad *symboltabell*. I (Figur 6) har användaren tryckt på F7 fyra gånger och ska vid nästa tryck exekvera beräkningen.



Figur 6

Modulotyp: Selektion (if)

Ett enkelt sekventiellt program exekverar alltid på samma sätt, men ibland vill en programkonstruktör att programmet skall göra ändra sitt instruktionsflöde beroende på vissa villkor, inmatade värden eller beräkningar. Detta kallas *selektion* eller på vanlig svenska för val. Vilket val programmet ska göra beror på villkoret som ställs. I datorsammanhang testas ett uttryck om det är sant (*true*) eller falskt (*false*). Det betyder att det bara finns två utgångar från en enkelt selektionsinstruktion. Flopp klarar följande tester

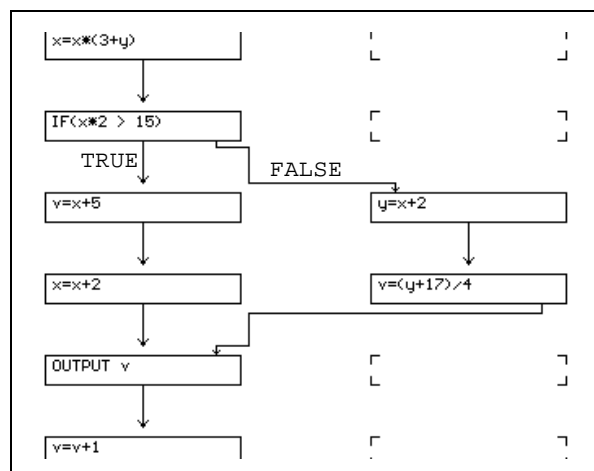
Symbol	Betydelse
==	lika med (obs dubbla '=')
!=	skilt från, ej lika med
>	större än
<	mindre än
>=	större än eller lika med
<=	mindre än eller lika med

I Flopp görs selektion genom att jämföra två uttryck och ur detta konstatera om jämförelsens påstående är sant eller falskt, se nedanstående exempel:

Förutsättning	Testuttryck	Betydelse	"Svar"
inga	$3 > 7$	3 är större än 7	falskt
foo=5	foo == 7	foo är lika med 7	falskt
foo=5, bar=5	foo == bar	foo är lika med bar	sant
foo=5	foo != 7	foo är skilt från 7	sant
foo=5, bar=5	foo != b	foo är skilt från bar	falskt
foo=5, bar=3	foo == bar+2	foo är lika med bar+2	sant

Flopp tillåter beräkningar (som läsaren kanske noterat) i IF-satser*. När flödespilar dras från en IF-modul frågar Flopp om flödet skall antas om det är sant (markeras grönt) eller falskt (markeras rött). Det går givetvis inte att ha två falska eller två sanna flödespilar från en IF-modul då tvetydighet skulle råda. I (Figur 7) kommer programmet göra olika beräkningar† beroende på om uttrycket i IF-modulen är sann eller falsk. Observera att uttrycket måste stå inom parentes.

Selektion kan dels användas till att välja alternativa vägar (framåt) i ett program men även *iterativt* (bakåt) i en s.k. *loop* för att repetera koden tills villkoret är uppfyllt.



Figur 7

Modulotyp: procedurer (proc, defproc & return)

I Flopp kan procedurer (kallas i vissa språk för funktioner) definieras och anropas. Vissa kodavsnitt (t.ex. en komplicerad beräkning) kanske förekommer fler gånger i ett program. Som programmerare vill man kanske modularisera sin program eller öka abstraktionen genom att anropa procedurer för att programmet ska vara lätt att följa, korrigera eller utöka. Risker för fel minskar också genom att det komplicerade avsnittet i programmet bara befinner sig på en plats istället för flera.

* men inte tilldelning av variabler i modulen

† följa olika vägar

En analogi till procedurer är matematikens funktioner. Att t.ex. använda polynomfunktionen $f(x) = x^2 + x - 3$, där x är inargument, är med gymnasiekompetens trivialt. $y = f(2)$ tilldelar y värdet 3. Abstraktionen $f(x)$ ser i de flestas ögon enklare ut än formeln $x^2 + x - 3$...

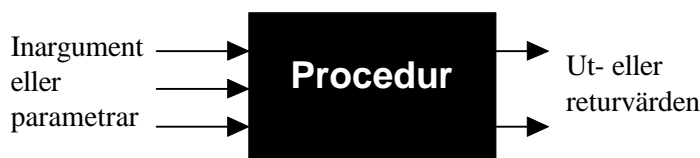
I Flopp anropas en egendefinerad procedur med

PROC funktionsnamn(inargument ,inargument,...,inargument: returvärde, returvärde,...,returvärde)

där antalet inargument (kallas även *parameter*) och returvärden är godtyckligt (dvs. kan även vara noll). Inargumenten kan vara ett godtyckligt uttryck*, emedan returvärden endast kan vara variabler.

Innan procedurer beskrivs ytterligare är det på sin plats att förklara hur *lokala variabler* fungerar.

Procedurer och matematiska funktioner kan ses som svarta lådor där man som användare bara behöver veta vad man ska stoppa in och vad man får ut:



Vad som händer inne i den svarta lådan är för användaren av lådan ointressant. Användaren bryr sig inte om hur beräkningarna utförs eller vilka algoritmer som används. Det som är intressant är att den gör det den ska och inte ändrar i några andra variabler än de som tilldelas returvärdena. Den svarta lådan får alltså inte påverka något annat i programmet. Detta innebär i förlängningen att de variabler som kanske kommer användas i lådan inte "läcker ut" till övriga delar av programmet. Inte heller ska konstruktören av själva lådan behöva ta hänsyn till vilka eventuella variabler som används i det "användande" programmet. Det innebär i sin tur att om lådkonstruktören använder en variabel inne i lådan som heter *FOO* inte påverkar en eventuell variabel med samma namn (*FOO*) utanför den svarta lådan. En variabel vid namn *FOO* ska följaktligen kunna finnas både i och utanför lådan utan att för den sakens skull vara samma sak eller ens kunna sammanblandas av programmet.

Till saken: En variabel som befinner sig inne i en procedur kallas lokal och påverkar inte livet utanför proceduren oavsett vilket namn den har. När proceduren avslutas och eventuella returvärden skickats tillbaka så förstörs alla lokala variabler och "försvinner". De finns inte ens kvar nästa gång proceduren anropas (används). Detta är ett mycket viktigt beteende som man som programmerare *måste* förstå

Procedurnamnet kan väljas godtyckligt enligt samma regler som variabelnamn. Anropet PROC namn... innehåller ett godtyckligt antal inargument och returvärden. Inargumenten fångas upp av DEFPROC namn (där namnet måste vara identiskt med det anropade för annars hittar inte Flopp den). Observera att inargumenten tilldelas de lokala variabler som finns uppställda i DEFPROC och att de måste vara lika många som i anropet. Exempelvis så innehåller variablerna $foo=5$ och $bar=3$ i ett program som sedan kommer till modulen

```
PROC calc_xy(foo,bar:x,y)
```

Flopp letar nu upp modulen med namnet

```
DEFPROC calc_xy(in1,in2)
```

och därmed innehåller $in1=5$ och $in2=3$. Observera att foo och bar inte kan komma å från proceduren, men de är inte förstörda utan de "lever" kvar i det "övriga" programmet (*mer avancerat*: mera exakt i en annan symboltabell vilket innebär att alla procedurer har en egen symboltabell där de lagrar sina lokala variabler).

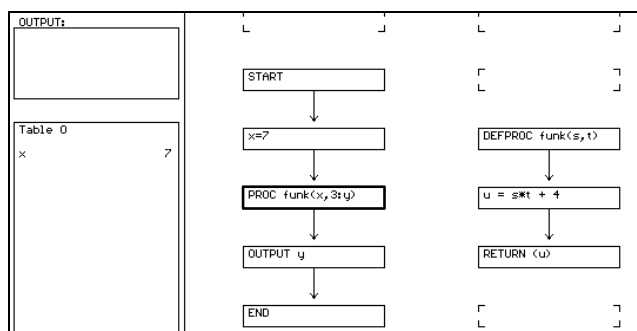
Om proceduren tar emot inparametrarna med

```
DEFPROC calc_xy(foo,bar)
```

eller

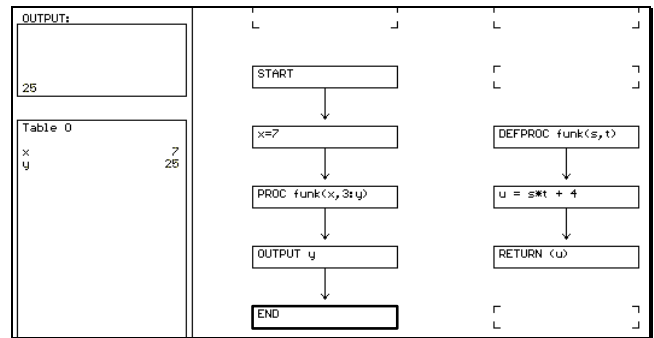
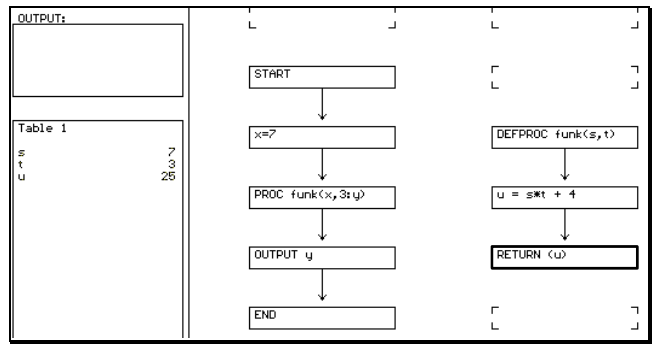
```
DEFPROC calc_xy(bar,foo)
```

så kommer dessa variabler att finnas i proceduren utan att påverka foo och bar -variablerna "utanför".



* d.v.s. kan vara variabel eller konstant

Nåväl, i slutet på proceduren vill programmeraren kanske skicka tillbaka ett eller flera returvärden och det görs i modulen RETURN(variabel1, variabel2) och dessa kommer på samma sätt som proceduranropet att tilldelas (i detta fallet) x och y. Liksom i anropsfallet så måste antalet returvärdet på "sändarsidan" vara lika många som på "mottagarsidan". I Figur 8 illustreras ett funktionsanrop i tre steg; strax innan proceduranropet, strax innan proceduranropet termineras och strax efter proceduren är avslutad.



Om du som användare hittar något fel, har andra önskemål gällande programmets beskaffenhet eller bara vill säga hur bra du tycker Flopp är så kontakta konstruktören som med glädje mottager all slags kritik.

Filip Sebek
 e-post: filip.sebek@mdh.se
 telefon: 021-103113