I IBM 7090 var NASAs allra första dator





IBM 7090 var en av de första transistordatorerna vilket gjorde den billigare och mindre(!) än de tidigare med elektronrör.

Programmen skrevs på hålkort som datorn läste av. Kortet nedan består av en rad kod!



Datan sparades på stora magnetband och resultatet av beräkningarna kunde skrivas ut på en skrivare.

IBM 7090 hade en beräkningshastighet på 100 KFLOP/s.

En iPhone 7 har en beräkningshastigheten på ungefär 300 GFLOP/s.

En iPhone 7 är alltså 3 miljoner gånger snabbare än IBM 7090 var.

Programmeringsspråket som de använde i filmen heter Fortran (Dorothy lånar/tar en bok om Fortran på biblioteket för att lära sig programmera datorn) och ni ska få testa att programmera i det.

Programmering i Fortran

1. Öppna upp länken under <http://husbynv.weebly.com> ->År 9 ->Teknik

 ->Länk Fortran

I editor rutan står följande:

program sum

 REAL X,Y,Z

 X = 10

 Y= 25

 Z = X + Y

 PRINT \*,"sum of x + y = ", Z

end program sum

1. Tryck på knappen ”Execute”. Vad händer?
2. Byt ut värdet på X och Y mot andra tal. Vad händer när du trycker på ”Execute” nu?
3. Om du istället vill utföra en subtraktion, multiplikation och division, vilka delar av koden behöver du ändra på och hur? Testa dig fram! Vad förstår du att datorn vet sedan innan?
4. Vad kan du dra för slutsatser om vad de olika delarna i koden används till?

Klistra in detta program i FORTRAN editorn istället:

program posneg

if (x .gt. 0) then

 print \*, "x is positive"

else if (x .lt. 0) then

 print \*, "x is negative"

 end if

end program posneg

1. Trycke på ”Execute”. Förklara varför inget händer. Vad behöver du lägga till?
2. Om du väljer värdet X=0 skrivs fortfarande inget ut. Kan du lägga till något i koden så att det står ”x is zero” om X=0? Ta hjälp av ordlistan och den kod som redan finns.

Liten Fortran-ordlista

.lt. less than

.gt. greater than

.le. less than or equal to

.ge. greater than or equal to

.eq. equals

.ne. not equal to

.and. and

.or. or

.not. not

.xor. "exclusive" or (i.e., only one is true)

.eqv. equivalent (i.e., same truth values)

.neqv. not equivalent