Særtryk fra Ingeniøren - nr. B 21 - 1. november 1962

GIER –
dagligt værktøj for Risø's medarbejdere

Af civilingeniør Helge Vilstrup,

Atomenergikommissionens Forsøgsanlæg, Risø

585.5:621.38

Programmeringen af elektroniske cifferregnemaskiner har udviklet sig fra at være en kunstart for specialister til at være noget, der kan indgå som et naturligt led i det daglige arbejde for enhver civilingeniør.

I nedenstående artikel gives en gennemgang af, hvorledes denne udvikling har fundet sted, og hvordan medarbejdere på Risø igennem 3-dages kurser bliver helt fortrolige med anvendelsen af elektronregnemaskinen GIER.

Indledning.

I det daglige arbejde på Risø er der et meget stort behov for at få løst beregningsproblemer af alle arter. En stor del af disse opgaver er af en så kompliceret natur, at de kun kan løses ved hjælp af en elektronisk cifferregnemaskine. Dette er baggrunden for Risøs anskaffelse af en elektronregnemaskine af typen GIER.

I forbindelse med anskaffelsen af GIER har der meldt sig en række problemer, såsom hvilket kodesprog man skulle anvende – om det skulle være den enkelte medarbejder, der skrev sine egne regnemaskineprogrammer – eller det skulle være en specialistgruppe, der skulle udføre dette arbejde, etc.

Erfaringer fra de tidligere år, hvor Risø har anvendt cifferregnemaskinen DASK, har vist, at kun de færreste civilingeniører har kendskab til, hvilke opgaver de med fordel kan løse ved hjælp af en cifferregnemaskine. Dette forhold er en afgørende årsag til, at en række problemer, der rettelig burde behandles på cifferregnemaskine, enten opgives som for komplicerede, løses på en tidsmæssig uøkonomisk måde eller løses under anvendelse af utilfredsstillende approximationer. Desuden har det vist sig, at det praktisk talt kun er de medarbejdere, der har kendskab til elektronregnemaskiner (for eks. ved at have fulgt et kodningskursus), som tænker på at løse deres problemer ved maskinel behandling.

Disse erfaringer er årsag til, at anvendelsen af GIER på Risø er organiseret på en lidt usædvanlig måde, hvor hovedformålet har været, at den enkelte medarbejder så let som overhovedet muligt skulle blive i stand til at skrive sine egne regnemaskineprogrammer og til at vurdere, hvorvidt det var økonomisk forsvarligt at programmere en given opgave til maskinel bearbejdning. Disse bestræbelser har fået et overraskende godt resultat, således at GIER idag står som et velkendt og magtfuldt værktøj for en stor del af Risøs medarbejdere. Da vore erfaringer med denne organisationsform må formodes at have interesse for en videre kreds, skal der i det følgende redegøres nærmere herfor. Denne redegørelse forudsætter ikke, at læseren har kendskab til elektronregnemaskiner.

Den elektroniske cifferregnemaskines egenskaber.

Cifferregnemaskinens væsentligste egenskab er dens enorme regnehastighed. På DASK og GIER regnemaskinerne er der tale om hastigheder af størrelsesordenen 1 million additioner pr. minut. Disse store regnehastigheder er muliggjort af, at maskinernes indre ikke indeholder mekaniske dele, men udelukkende er baseret på elektroniske komponenter. Medens DASK, der blev bygget i 1957 af Regnecentralen, i det væsentlige er baseret på radiorør, er GIER, Regnecentralens nye maskine, udelukkende opbygget på grundlag af transistorer og dioder.

Elektronregnemaskiner indeholder følgende grundenheder:

Læse- og skriveenheder, der forbinder maskinen med omverdenen,

hukommelse, hvor tal og ordrer kan gemmes, samt en

aritmetisk enhed, hvori de egentlige beregninger foregår.

Ser man på forskellige regnemaskinetyper, vil man bemærke, at disse grundenheder kan være af meget forskellig natur og ydedygtighed. I maskiner konstrueret til teknisk-videnskabelige beregninger lægges der sjældent så stor vægt på læse- og skriveudstyrets hastighed, hvorimod det er afgørende, at den aritmetiske enhed har så stor regnehastighed som muligt, idet denne type opgaver i reglen involverer overordentlig omfattende numeriske beregninger, mens ind- og udlæsning af data spiller en ret ringe rolle. Ser man derimod på elektroniske databehandlingsanlæg, som er navnet på de cifferregnemaskiner, der anvendes til kontorautomationsformål, finder man, at der i disse anlæg er lagt meget stor vægt på, at læse- og skriveenhederne virker så hurtigt som muligt, hvorimod den aritmetiske enheds regnehastighed ofte er relativt lav, idet kontorautomationsproblemer indeholder relativt få aritmetiske operationer.

GIER er velegnet til teknisk-videnskabelige formål med sin ret høje regnehastighed. Eksempelvis udføres

1 W

en addition af to tal med 9–12 decimale cifre på 22 til 75 μ sek., afhængigt af tallenes art, mens en multiplikation udføres på 150 μ sek.

Ind- og udlæsningen af information til og fra maskinen kan ske ved hjælp af hulstrimler. I fig. 1 ses et nærbillede af perforatoren, der kan hulle resultatstrimler med en hastighed af 150 tegn/sek. svarende til 40 cm/sek. En hulrække på tværs af hulstrimmelen svarer til eet tegn, bogstav eller ciffer. Hulstrimler, der skal læses af GIER, er fremkommet som et biprodukt fra en elektrisk skrivemaskine, en såkaldt Flexowriter, og hulsymbolerne korresponderer nøjagtigt med det, der er blevet skrevet på skrivemaskinen. Hulstrimler fra GIER med resultater indsættes blot i Flexowriteren, der derefter automatisk skriver resultaterne på papir.

På fig. 2 ses et fotografi af Risøs GIER-installation, hvor man til venstre kan se hulstrimmellæseren, der kan læse 500 tegn pr. sek. Bag perforatoren, til højre i billedet, ses den såkaldte kontrolskrivemaskine, på hvilken man kan lade resultaterne udskrive – eller skrive ind i GIER. Som navnet antyder, anvendes denne skrivemaskine normalt til kontrolformål, idet anvendelse af hulstrimler er langt hurtigere. Bagest i billedet kan man se ind i GIER gennem den åbne dør. Foroven ses de udskiftelige plader, på hvilke elektronikken er opbygget, og forneden den roterende magnetiske tromle, der fungerer som en del af hukommelsen. På bordet i midten står kontrolpulten, hvorfra man har adgang til regnemaskinens hukommelse og registre. (1).

Da elektronregnemaskinen er konstrueret til at give store regnehastigheder, kan man ikke lade en operatør trykke på knapper og skrivemaskinetaster, mens beregningerne står på, alt må ske automatisk – og man har derfor måttet finde en anden metode til instruktion af maskinen. Instruktioner, der fortæller regnemaskinen, hvad den skal gøre, kaldes for ordrer, og en række af sådanne ordrer danner et regnemaskineprogram.

Ordrerne lagres i maskinens hukommelse på samme måde som tal, d. v. s. i 2 talssystemet, idet dette talsystem naturligt kommer i anvendelse i forbindelse med de elektroniske kredsløb, der mest driftsikkert kan være i een ud af to tilstande, som f. eks. et radiorør, der enten trækker strøm eller ikke trækker strøm. Set fra et forbrugersynspunkt er det ikke væsentligt, at maskinen arbejder i det binære talsystem, idet dens hukommelse indeholder små programmer, der kan læse decimale tal eller ordrer i en given notation og omregne informationen til det binære talsystem, således at den kan lagres i hukommelsen. Ligeledes indeholder hukommelsen udlæseprogrammer, der kan udføre den modsatte funktion.

Elektronregnemaskinens ordrer er af meget simpel karakter. F. eks. kunne en ordre være: »Tag et tal fra en given hukommelsescelle og adder det til det tal, der står i et bestemt regneregister« eller »læs et tegn fra en hulstrimmel« o. s. v. De funktioner, der er inkluderet blandt de normale ordrer vil være: addition, subtraktion, multiplikation, division, logiske operationer og lignende. De enkelte ordrer er således af en simpel og grundlæggende natur, hvilket er forklaringen på, at

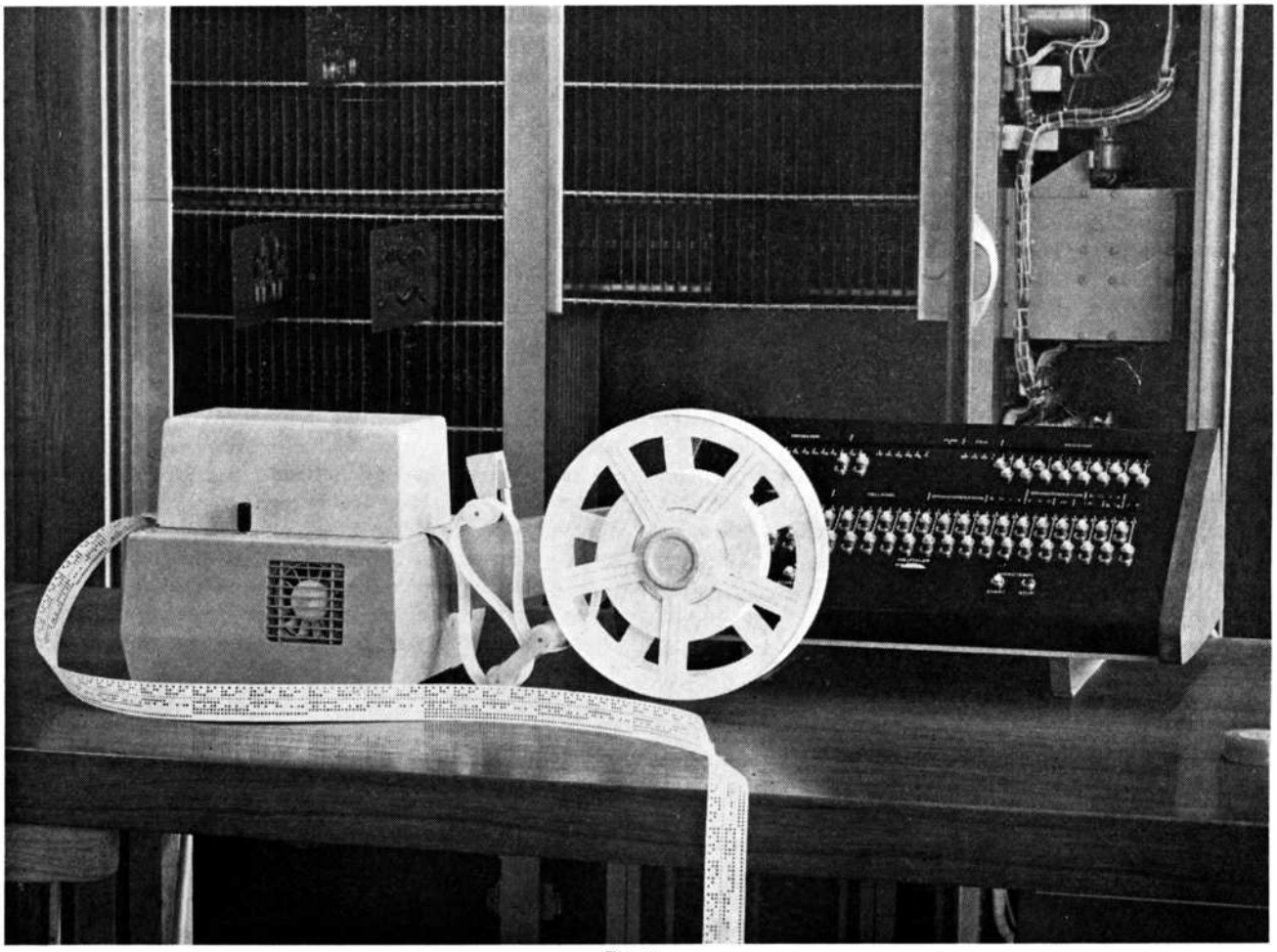

Fig. 1.

elektroniske cifferregnemaskiner kan løse opgaver af vidt forskellig karakter – fra egentlige beregningsopgaver, baseret på vilkårlige numeriske metoder – til sprogoversættelse – beregning og tegning af diagrammer for nye elektronregnemaskiner – og komposition af musikstykker.

Som et eksempel på hvordan ordrekoden for henholdsvis DASK og GIER ser ud i praksis, kan vi beregne:

$$A = B + C$$

hvor vi forudsætter, at B og C er lagret i hukommelsescelle nr. 100 og 102, og at resultatet A skal anbringes i celle 104. I DASK maskinkode vil beregningen se således ud:

```
100 A 40
2021 A 16
102 A 40
2026 A 16
2 A916
2016 A 16
104 A 08
```

altså ingenlunde opmuntrende for den, der har lyst til at skrive sine egne regnemaskineprogrammer. På GIER vil beregningen se ud på følgende måde (2):

```
ARSF 100
ARF 102
GRF 104
```

altså allerede en noget simplere struktur. Det er ligefrem læseligt, at man skal tage celle nr. 100 og *A*ddere den til *R*esultatregisteret, der er *S*lettet; *A*ddere celle nr. 102 til *R*esultatregisteret og *G*emme *R*esultatregisteret i celle 104. På den anden side er det unægtelig en vanskelig måde at definere ovenstående simple formel på. Man ser, at der er rige muligheder for fejltagelser. I praksis viser det sig også, at det er meget tidsrøvende at skrive regnemaskineprogrammer i maskinens eget sprog, sådan som det er prøvet i ovenstående eksempler – yderligere tager det lang tid at finde fejlene i disse programmer. Hertil kommer, at det kan være næsten umuligt for andre end forfatteren at læse og forstå et sådant program, ligesom det er meget besværligt, at give en ordentlig beskrivelse af programmet.

Der har på et tidligt tidspunkt været arbejdet med spørgsmålet, om ikke man kunne finde frem til et kodesprog, der lå den menneskelige natur nærmere. Som vi

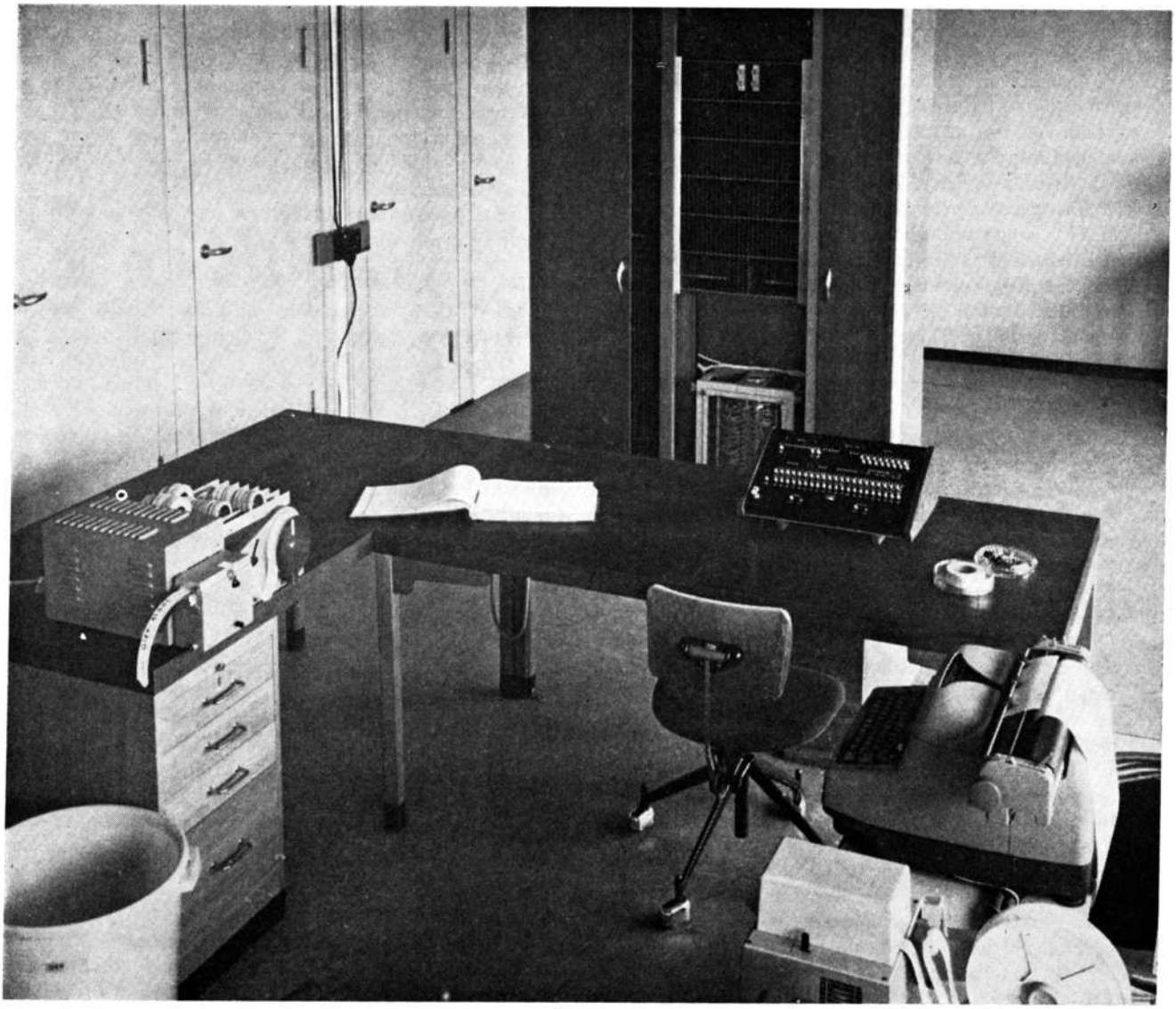

Fig. 2.

ser af ovenstående eksempel, blev vort problem jo næsten helt defineret i udtrykket A = B + C, og det ville derfor være naturligt at forlange, at man kunne skrive sine programmer i et sprog, der mindede stærkt om den sædvanlige matematiske notation – og så enten lade regnemaskinen oversætte dette formelsprog til sit eget maskinsprog (det er muligt, fordi regnemaskinen kan udføre logiske processer) – eller også opbygge regnemaskinerne således, at de direkte kan forstå formelsproget. Den sidste løsning har af flere grunde ikke så stor tiltrækning, hvorimod ideen med at lade maskinen oversætte et formelsprog er blevet ført ud i praksis allerede for adskillige år siden. Som det mest kendte eksempel på et sådant tidligt udviklet formelsprog, har vi IBM's FORTRAN (FORmular TRANslation), brugt overordentlig meget på mange IBM-maskiner (3). Imidlertid var dette sprog på en række punkter utilfredsstillende, det lå stadig for tæt (set fra et brugersynspunkt) ved maskinernes eget sprog og har derfor ikke resulteret i en udvikling af maskinerne.

En række andre sprog af tilsvarende karakter er dukket op, men en egentlig nyskabelse er først opnået i 1960, ved definitionen af det internationale sprog ALGOL 60, udviklet af en komité bestående af 7 europæere og 7 amerikanere. I dette samarbejde har Regnecentralen, ATV, ydet meget vægtige bidrag gennem komiteens sekretær, dr. phil. Peter Naur og civilingeniør Jørn Jensen, begge fra Regnecentralen. Dette er formodentlig årsagen til, at Danmark i dag er placeret overordentlig fint i arbejdet på at udnytte ALGOL. (4).

At behovet for at få et effektivt, brugerorienteret regnemaskinesprog har meldt sig med så stor styrke netop i disse år skyldes, at elektronregnemaskinerne er blevet hurtigere, større og billigere, samtidig med at kravet om programmering af nye opgaver er vokset exponentielt. Det er derfor naturligt, at brugernes ønsker om let programmerbare regnemaskiner har vundet terræn, idet det nu er økonomisk forsvarligt at lade regnemaskinerne overtage en stadig større del af programmeringsarbejdet.

Egenskaberne ved ALGOL 60.

Det væsentlige nye ved ALGOL kontra tidligere regnemaskinesprog er, at ALGOL er konstrueret udfra et forbrugersynspunkt, så nær op ad den normale matematiske notation som muligt, medens tidligere regnemaskinesprog var konstruerede med nøje henblik på, hvorledes de aktuelle maskiner fungerede. De ældre regnemaskinesprog har derfor været relativt ulæselige, mens man har lagt stor vægt på at gøre ALGOL så let læseligt som muligt. ALGOL tjener derfor nu 2 formål. Det ene er at være basis for udveksling af beskrivelse af regneprocesser, og det andet er den egentlige programmering af elektronregnemaskiner. Som et eksempel på den første anvendelse kan det nævnes, at en række tidsskrifter til stadighed offentliggør algoritmer, og at det f. eks. på Moskvas universitet forlanges, at alle beskrivelser af numeriske metoder skal være affattet i ALGOL. Med henblik på anvendelsen af ALGOL som regnemaskinesprog har der i starten blandt mange været nogen usikkerhed, om hvorvidt det virkelig var muligt at skrive gode oversætteprogrammer til dette sprog, men det kan allerede nu ses, at det meget vel er muligt. Som et eksempel herpå har vi DASK og GIER ALGOL-oversætterne (5, 8), samt oversættere til en række andre maskiner i Europa og USA.

Ved definitionen af de symboler, der bør anvendes i ALGOL 60, har man ikke taget hensyn til det apparatur, der normalt eksisterede i forbindelse med elektronregnemaskinerne, men først og fremmest til at sproget skulle være læseligt. Dette har allerede fået betydning for modificering af regneanlæggene henimod forbrugerønskerne. Således er DASK og GIER udstyret med læse- og skriveapparater, der nøje følger ALGOL's definitioner.

Det er ikke hensigten på dette sted at give nogen detaljeret gennemgang af ALGOL, idet der f. eks. kan henvises til en udmærket oversigtsartikel (6) eller en lærebog i hele sproget (7). På den anden side skal der gives en grov skitse af sprogets egenskaber af hensyn til forståelsen af resten af artiklen.

Da vi ovenfor ville beregne udtrykket A = B + C, sagde vi, at beregningen var næsten defineret ved denne formel. Det er en typisk egenskab ved den normale matematiske notation, at definitioner af regneprocesser sker ved en blanding af formler og tekst. Ser vi på formlen

$$A = B + C,$$

kan vi ikke vide, om den betyder »beregn A af B + C« eller om den definerer et spørgsmål »om A er lig med B + C«. I ALGOL har man derfor vedtaget, at beregningsudtrykket skal skrives som

$$A := B + C;$$

samtidig forlanges der en forklaring om, hvad de navne, man benytter i ALGOL programmet, betyder. A må f. eks. være navnet på en variabel størrelse, der gennem denne formel opnår en talværdi, medens B kunne være en variabel (med en given talværdi) eller en funktion af visse størrelser.

ALGOL-sproget indeholder en række standardfunktioner. F. eks. kunne vi skrive

$$A := \sin(B) + \cos(C) - 0.34 \times B;$$

Vore ALGOL-formler kan også indeholde elementer af vektorer, matricer eller flerdimensionale talsæt. Element nr. 9 i vektor D skrives således D[9], idet index af praktiske grunde er rykket op på linien. Foruden de aritmetiske operatorer indeholder sproget en række særlige ALGOL-operatorer, der har til formål at lette regningen med matricer, gøre sproget entydigt og indføre de egenskaber, der er karakteristiske for løsningen af opgaver på elektronregnemaskiner. Udover den normale aritmetik indeholder sproget også en fuldstændig, logisk algebra.

ALGOL er opbygget overordentlig generelt. I modsætning til de ældre regnemaskinesprog, findes der meget få undtagelser fra reglerne, og sproget er klart grammatisk og logisk defineret. Ved mange tidligere regnemaskinesprog stod det således til tider brugeren uklart, om en given sætningsopbygning var tilladt i det pågældende sprog, og svaret kunne ikke findes i håndbogen for dette. Lignende problemer findes praktisk talt ikke i ALGOL.

En meget væsentlig egenskab ved ALGOL er, at sproget tillader, at man selv definerer nye funktioner udover standardfunktionerne. Hvis vi f. eks. har brug for at beregne

$$u! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots\ldots \cdot (u-1) \cdot u$$

flere steder i et program eller i flere forskellige programmer, kan vi een gang for alle skrive et lille stykke ALGOL-program, en såkaldt funktionserklæring, der

fortæller, at et bestemt navn f. eks. »fakultet« skal referere til ovenstående beregningsforskrift. I de programmer hvor fakultetsfunktionen skal beregnes, indføjes blot den pågældende funktionserklæring, og vi kan nu beregne fakultetsfunktionen, nøjagtig som om den var en standardfunktion. Skal vi f. eks. beregne alfa som

$$\text{alfa} = \binom{u}{k} = \frac{u!}{k! \cdot (u-k)!}$$

kan dette skrives som

```
alfa := fakultet(u)/( fakultet(k) × fakultet(u - k) );
```

Organisation af GIER's anvendelse på Risø.

I arbejdet med Risø-GIER tilstræbes det at tilfredsstille følgende krav:

1. Sikre at de enkelte medarbejdere har så godt kendskab til GIER, at de kan vurdere, hvorvidt et givet problem bør programmeres.
2. Sikre at opgaverne kan kodes uden ventetid, således at denne ikke bliver anledning til, at velegnede opgaver ikke bliver maskinelt behandlet.
3. Sikre at opgaverne bliver programmeret på den bedst mulige måde.
4. Gøre anvendelsen af GIER så enkel og hurtig som mulig.
5. Sikre at der fremstilles en klar beskrivelse af ethvert regnemaskineprogram.
6. Undgå at flere medarbejdere, hinanden uafvidende arbejder med kodning af det samme problem, eller at en medarbejder programmerer en opgave, der allerede er løst på Risø eller på en anden GIER-regnemaskine.

Valget af programmeringssprog er i høj grad bestemmende for, om disse mål kan nås. Der skal derfor først redegøres for, hvordan man har valgt mellem de forskellige muligheder.

Valg af programmeringssprog.

Det er svært at bedømme, hvor meget det koster at bearbejde problemerne maskinelt. Det, der skinner mest i øjnene, er den regnemaskinetid, der benyttes, og man vil let være tilbøjelig til at sammenligne udgiften til regnemaskinetid med udgiften til en tilsvarende håndregning, hvis en sådan overhovedet kan gennemføres. Dette er imidlertid ikke relevant. Praksis viser, at programmerings- og fejlfindingstiden udgør en meget væsentlig del af budgettet.

På en forskningsinstitution må man forvente, at programmernes levetid er relativ kort – formler forbedres, forsøgsopstillinger ændres, nye opgaver dukker ustandselig op – og man vil således kunne forvente, at programmeringsbyrden her vil blive særlig stor.

Ved vurderingen af, om man skal benytte ALGOL eller maskinsprog, viser det sig, at ALGOL har en række fordele frem for maskinsproget, men også to uheldige egenskaber. ALGOL-programmerne vil regne langsommere end maskinprogrammerne, og de vil fylde mere af maskinens hukommelse. Den sidste egenskab er i øjeblikket kun af betydning for en lille procentdel af Risø's opgaver, mens den første ulempe naturligvis får indflydelse på samtlige programmer.

Erfaringer fra DASK har imidlertid vist os, at det praktisk talt i alle tilfælde er økonomisk fordelagtigt at benytte ALGOL, og dette forhold synes at blive endnu mere udtalt på GIER, selvom vi her endnu mangler et passende erfaringsmateriale.

Hertil kommer, at ALGOL har en hel række fordele, hvis økonomiske betydning ikke kan beregnes, og som ikke indgår i ovenstående vurdering. Disse egenskaber vil tildels fremgå af det følgende. Her skal blot nævnes, at vi har kunnet skrive vore programmer i ALGOL og benytte disse på DASK, indtil GIER–ALGOL oversætteren nu er blevet færdig, således at vi kan flytte 50–60 programmer over på GIER uden nogen form for ændring. Dette til trods for at DASK og GIER har totalt forskellige ordresystemer.

Regnemaskinegruppens opgaver.

Til at sikre en effektiv drift af GIER, blev der oprettet en lille specialistgruppe i forbindelse med maskinen. Gruppen består af 6 personer: to assistenter, der passer Flexowriterne, to civilingeniører, hvoraf den ene er særlig kendt med ALGOL-sproget, og den anden har et nøje kendskab til mikroopbygningen af maskinens ordrer, en ingeniør, der har været med til at bygge maskinen, og som har det tekniske ansvar, foruden at han fungerer som operatør, og endelig en magister i matematik, der hjælper ved løsningen af matematiske og numeriske problemer og giver Risø's bibliotek støtte i forbindelse med anskaffelse og katalogisering af matematisk litteratur.

Regnemaskinegruppens opgave er at gøre anvendelsen af GIER så let og effektiv som mulig for Risø's medarbejdere. Denne opgave har vi søgt at løse ved at sætte ind på følgende punkter:

1. Undervisning.

Da arbejdet skulle baseres på ALGOL i så vid udstrækning som muligt, var den tanke nærliggende, at bibringe alle interesserede medarbejdere et sådant kendskab til ALGOL, at de selv kunne skrive deres programmer. Der blev derfor afholdt et forsøgskursus for 2 medarbejdere, som fik et godt kendskab til hele ALGOL-sproget. Det viste sig, at kursus kom til at vare i 3 uger, og at eleverne var ret usikre, når de skulle benytte deres viden, fordi de havde så mange forskellige programmeringsmuligheder at vælge imellem i ALGOL. Den største vanskelighed ved kursus syntes at være, at alt i sproget hænger så nøje sammen, at det ikke er muligt at lære en ting til bunds uden at kende resten af sproget.

Da det vil være utænkeligt at tage medarbejdere ud af det daglige arbejde i 3 uger, for at de kan lære GIER og ALGOL at kende, blev det besluttet at udarbejde et ALGOL-kursus, som kunne gennemgås på 3 dage. Grundideen i dette kursus er at fjerne alle de dele af ALGOL, der ikke indgår i teknisk-videnskabelige beregninger, og forenkle sproget, så kun en af flere lige gode programmeringsmuligheder læres, idet hovedvægten lægges på at gøre programmerne let læselige og nedsætte fejlhyppigheden. Endvidere baseres indlæringen af sproget på »naturmetoden«, således at man på intet tidspunkt lærer noget, der forudsætter kendskab til emner, der endnu ikke er gennemgået. I forbindelse med det sidste punkt har det arbejde, som dr. techn. J Kjær har udført hos firma Haldor Topsøe, været til stor støtte. Erfaringen med dette nye kursus, der først blev prøvet for 8 deltagere, viste at tidsfristen stort set kunne overholdes, idet det var tilstrækkeligt at undervise i 3 dage à 5 timer, mens yderligere $^1/_2$ dag blev

benyttet til gennemgang af en række hjemmeopgaver. I løbet af disse 3 dage læres ikke blot ALGOL-sproget, men også en række praktiske ting i forbindelse med udarbejdelsen af regnemaskineprogrammer. Desuden behandles forskellige numeriske problemer.

Det må understreges, at deltagerne trods undervisningens kortvarighed får et grundigt kendskab til den del af ALGOL-sproget, som kursus omfatter, idet de fleste ting gennemgås 2 à 3 gange. Da desuden deltagerantallet holdes lavt (8–15 personer), bliver undervisningen effektiv, idet de enkelte kan komme frem med deres spørgsmål og få dem indgående besvaret.

Man kan nu spørge: hvor meget er dette begrænsede ALGOL værd. Det har her vist sig, at de, der har lært ALGOL i dets fulde udstrækning, i hovedsagen ikke bruger andet og mere end det, der anvendes i det begrænsede ALGOL, og i de tilfælde, hvor de gør det, kunne de med lige så stor fordel have anvendt udtryk, der fremkommer i det forkortede ALGOL. Yderligere synes det, som om de medarbejdere, der har lært det begrænsede ALGOL er mere sikre i deres anvendelse af sproget umiddelbart efter endt undervisning, end deltagere i det første forsøgskursus, der lærte ALGOL i dets fulde udstrækning.

En relativt stor del af undervisningstiden anvendes til en nøje gennemgang af de såkaldte funktionserklæringer, der er noget af det vigtigste, men også noget af det sværeste i ALGOL, således at deltagerne opnår at forstå dette begreb til bunds.

En egenskab, der ikke læres i ALGOL-kursus på Risø, er logisk algebra, idet de færreste civilingeniører har kendskab til eller brug for denne. De få, der skal bruge logisk algebra i forbindelse med f. eks. beregning af relækredsløb, kan hurtigt lære den del af ALGOL, der omhandler denne, når de iøvrigt er fortrolige med ALGOL.

2. Udarbejdelsen af programmer.

Når en medarbejder ønsker at løse et problem ved hjælp af GIER, vil han først undersøge en liste med resumeer af samtlige programmer, for at se om hans problem allerede skulle være helt eller delvis løst. Er dette ikke tilfældet, vil han gennem en henvendelse til regnemaskinegruppen få oplyst, om andre på Risø er i gang med at løse det pågældende problem, om det er løst på andre GIER-regnemaskiner, eller om det eventuelt er løst på regnemaskiner af helt andre typer.

Hvis opgaven skal kodes til GIER, vil regnemaskinegruppen assistere med løsning af eventuelle matematiske og kodningsmæssige problemer, og såfremt visse dele af opgaven har almen interesse, kan regnemaskinegruppen overtage programmeringen af disse. De vil i så fald blive programmeret som biblioteksfunktioner. En biblioteksfunktion er nøje afprøvet og kan indgå i ethvert ALGOL-program. Som eksempel på sådanne biblioteksfunktioner kan nævnes løsning af lineære ligninger, mindste kvadrat approximation af målepunkter med polynomier, beregning af Besselfunktioner, etc.

ALGOL-programmet kan nu nedskrives under anvendelse af biblioteksfunktioner til de dele af programmet, der er af generel natur. På denne måde vil programmeringsarbejdet koste mindst mulig ulejlighed, og der vil fremkomme færre fejl, da en del af programmet i forvejen er gennemprøvet.

Som et eksempel på nedskrivning af et ALGOL-program under anvendelse af en biblioteksfunktion kan vi beregne integralet

$$\int_{p\pi}^{(p+1)\pi} \frac{x \sin(x)}{(100 + x^2)^2}\, dx$$

idet vi ønsker at beregne integralet i ethvert af de områder, der ligger helt på den ene eller den anden side af x-aksen fra 0 op til 11π. ALGOL-programmet ser ud på følgende måde:

```
begin
real a, x, pi;   integer p;
comment library Simpson 2;

pi := 3.14159;
for p:= 0 step 1 until 10 do
begin
a := Simpson 2(x×sin(x)/(100+x↑2)↑2 , x , p×pi ,
                (1+p)×pi, 10-4 );
tryk vr;
tryk(≮n.dddd10d≯, a);
end of integralberegning;
end of program;
```

I de første linier står der, at a, x og pi er reelle tal, medens navnet p skal anvendes for et heltal. I næste linie står der en kommentar: at biblioteksfunktionen Simpson 2 skal indsættes på dette sted i programmet. Denne indsættelse sker i øjeblikket ved kopiering af den aktuelle hulstrimmel, men vil senere ske på en mere automatisk måde. I den følgende linie får pi sin talværdi, og vi går nu over til de egentlige beregninger.

Næste linie indeholder en ALGOL-operator, hvis betydning er, at p skal antage enhver af talværdierne 0, 1, 2 ....., 9, 10, og for enhver af disse værdier udføre det følgende stykke ALGOL-program mellem »**begin**« og »**end** of integralberegning«.

Dette program indeholder først en linie, hvor a får den talværdi, der beregnes med Simpson 2 funktionen. Læser man i beskrivelsen af denne biblioteksfunktion, ser man, at der som første parameter (d. v. s. hen til første komma) skal angives, hvilket aritmetisk udtryk man ønsker at integrere. Som anden parameter angives navnet på den uafhængige variable (her x), som tredie parameter nedre grænse for argumentet, som fjerde parameter den øvre grænse, og den sidste parameter angiver, hvilken relativ nøjagtighed man ønsker for resultatet, altså i dette tilfælde $1 \cdot 10^{-4}$.

Hvis vi sætter p = 0, ser vi, at nedre grænse er 0 og øvre grænse pi, og integrationen udføres nu for disse grænser. Når det første integral, altså p = 0, er blevet beregnet, fortsætter ALGOL-programmet i næste linie, hvor der står tryk vr. Det betyder tryk vogn retur, d. v. s. før papiret tilbage og skift til en ny linie. I næste linie trykkes talværdien for a med 5 cifre og komma efter første ciffer, samt eventuelt en 10-tals-exponent. Nu fortsætter programmet med at integrere med p sat lig 1 o. s. v., indtil den sidste integration er udført for p = 10, hvorefter maskinen passerer *end* of program, og stopper – klar til næste ALGOL-program. I virkeligheden kunne ALGOL-programmet være skrevet i een eneste sætning, men for at øge læseligheden er det her skrevet på den angivne måde.

3. Fejlfinding i nye programmer.

Når manuskriptet til ALGOL-programmet er færdigt, afleveres det til regnemaskinegruppen, hvor det bliver skrevet på Flexowriter, således at programmet nu foreligger på hulstrimmel. Da der erfaringsmæssigt altid vil

være grammatiske fejl i ALGOL-programmer, og det er af afgørende tidsmæssig betydning, at alle disse fejl så vidt muligt findes på een gang, gennemlæses hulstrimmelen nu ved hjælp af et ALGOL-testprogram på GIER.

Testprogrammet er konstrueret således, at det giver en udskrift på kontrolskrivemaskinen, hver gang det konstaterer en grammatisk fejl. Udskriften indeholder en klar definition af fejlens art og placering i programmet. Testprogrammet kan også undersøge, hvad der om fornødent skal tilføjes til et givet program, for at det i grammatisk forstand kan udgøre et afsluttet program.

Manuskript, Flexowriter-udskrift og udskriften fra testprogrammet returneres nu til forfatteren, der læser korrektur og retter fejl. En ny hulstrimmel fremstilles ved kopiering af den gamle, idet fejlene samtidig rettes, og som oftest vil denne nye strimmel være grammatisk korrekt, således at den kan oversættes, og kontrol **af** beregningsgangen kan begynde.

Denne kontrol, der har til formål at give sikkerhed for, at man virkelig har skrevet de rigtige formler etc., udføres almindeligvis uden særlige vanskeligheder. Hvis det under indkørsel af et program volder kvaler at finde årsagen til en eller anden fejl, kan regnemaskinegruppen som oftest hurtigt løse problemet, takket være at ALGOL er så let læseligt.

4. Programbeskrivelse.

Når arbejdet med at fremstille et program begynder, afleveres et resumé af programmets egenskaber til regnemaskinegruppen. Dette resumé udgives i en fælles abstractliste over samtlige programmer. Når arbejdet er afsluttet, udarbejder forfatteren en beskrivelse af programmet med oplysninger om dettes egenskaber.

En væsentlig del af programbeskrivelsen vil være selve ALGOL-programmet, der skrives på stencil af Flexowriteren ud fra den korrekte hulstrimmel. Der lægges stor vægt på, at der udarbejdes en beskrivelse for hvert eneste program, da de erfaringer, der er opsummeret i programmet, ellers stort set må betragtes som spildte.

Programbeskrivelsen udarbejdes som oftest på Flexowriter, således at der findes en hulstrimmel svarende til beskrivelsen. Hvis programmet senere bliver ændret, vil det således være meget let at ændre beskrivelsen ved delvis kopiering af den gamle strimmel. Samtidig opnås det, at genoptrykning af en særlig interessant beskrivelse kan udføres automatisk.

5. Rutinekørsel.

Som oftest vil programmerne være formuleret helt generelt, således at de kan benyttes til at løse alle problemer af den pågældende art. Som eksempel kan angives et program, der beregner neutronflux i en cylinderformet reaktor. Programmet er helt generelt formuleret og kan beregne flux i alle cylinderformede reaktorer, idet man blot må opgive talkonstanter, der definerer de geometriske forhold og sammensætningen af materialer.

I forbindelse med de fleste programmer bliver der derfor udarbejdet et skema, i hvilket man indfører de parametre, der definerer den aktuelle beregning. Talkonstanterne hulles på Flexowriteren på grundlag af skemaet, og beregningerne foretages nu, idet regnemaskinen først læser det pågældende program, og dette program beordrer indlæsning af parametrene, hvorefter maskinen udfører de aktuelle beregninger og afleverer resultater på en hulstrimmel.

Denne hulstrimmel indsættes i Flexowriteren, der automatisk udskriver resultaterne på papir i sammenhængende ark. Da alle ALGOL-programmer på Risø indeholder den samme biblioteksalgoritme, der sørger for trykning af overskrift på hver side samt automatisk sideskifte og desuden tiener til at standardisere en række andre forhold i forbindelse med udskriften, kan alle resultatstrimler behandles på samme måde og fuldstændig automatisk.

Konklusion.

Gennem det forløbne år, hvor vi har arbejdet på at gøre brugen af GIER almindelig for Risø's medarbejdere, har det vist sig, at regnemaskinen er en meget stor stimulans i arbejdet. Vi har gentagne gange set, hvorledes en gruppe medarbejderes arbejde har stået på et bestemt videnskabeligt niveau, indtil de pågældende beregningsopgaver blev programmeret. Først når belastningen fra beregningsarbejdet er fjernet, får de tid og kraft til at arbejde sig dybere ind i problemerne. På den anden side skal elektronregnemaskinen naturligvis benyttes med varsomhed, således at man ikke lader sig friste til en maskinel bearbejdning af de opgaver, der kan løses langt elegantere ved analytisk tankevirksomhed.

Litteratur:

(1) H. Isakson, B. Scharøe Petersen: GIER, systemplanlægning og kredsløb. Ingeniøren, 15. decbr. 1962, p. 721.
(2) T. Krarup, B. Svejgaard: GIER, den logiske struktur. Ingeniøren, 15. decbr. 1961, p. 716.
(3) Chr. F. Rovsing: Hurtig programmering af elektronregnemaskiner, FORTRAN. Ingeniøren, 1. juli 1961, p. 398.
(4) Peter Naur: Report on the Algorithmic Language ALGOL 60. Acta Polytechnica Scandinavia, MA 5, (284/1960).
(5) J. Jensen, T. Jensen, P. Mondrup, P. Naur: A Manual of the DASK ALGOL Language. Regnecentralen, 1961.
(6) Peter Naur: ALGOL — det internationale sprog til at beskrive logiske og numeriske processer. Nordisk Matematisk Tidsskrift, Vol. 8, p. 117.
(7) Chr. Andersen: Lærebog i ALGOL. Regnecentralen, 1961.
(8) Et nyt barn fra Regnecentralen: Ingeniøren Ugeblad, 22. septbr. 1962, p. 10.