Kapitel 1. Inledning

Vetenskapliga beräkningar, eller beräkningsvetenskap, handlar om konsten att använda räkneapparatur och metoder för att lösa vetenskapliga problem. Numera är det närmast fråga om datorer och metoder anpassade till dem, men många av de metoder, som numera används i beräkningsvetenskapen kom till långt före datoreran.

1.1. Datorernas historia

Redan den experimentella fysikens och astronomins utveckling på 1600–talet påverkades i hög grad av uppfinningen av många hjälpmedel som underlättade beräkningarna. Logaritmtabellerna som uppfanns på 1600–talet var av stor betydelse långt in på 1900–talet. De underlättade framförallt komplicerade aritmetiska beräkningar, eftersom multiplikation kunde ersättas av addition. Den första tabellen som uppgjordes av *John Napier* år 1614, innehöll 90 sidor tabeller: *Mirifici Logarithmorum Canonis descriptio* (ungefär "beskrivning av de underbara logaritmtabellerna").

Napier använde naturliga logaritmer, och därför blev Briggs logaritmtabell, som använde basen 10 populärare. Hans tabell publicerades 1624 och innehöll logaritmer för talen 1–20000 och 90000–100000 med 14 decimalers noggrannhet (varav dock 1161 eller 4 % innehöll fel). *Johannes Kepler* gjorde upp egna tabeller samma år och 1628 gav holländaren *Adrian Vlacq* ut en tabell över logaritmerna för talen 1-101000, som kom att användas under lång tid framöver. Man anser att få uppfinningar har underlättat vetenskapliga beräkningar så mycket som logaritmtabellerna gjorde (med undantag av datorerna givetvis).

Räknestickorna, som bygger på logaritmerna, har sitt ursprung i tabellerna. Den första räknestickan var antagligen den engelska matematikern Edmund Gunter's *line of numbers*, ett stycke trä, några fot långt, där en logaritmisk skala blivit markerad. En annan engelsk matematiker, *William Oughtred*, som också verkade som präst, förbättrade räknestickan, så den blev praktiskt användbar.

Den första mekaniska räknemaskinen konstruerades av *Wilhelm Schickard* ca 1623. Han var professor i orientaliska språk och matematik samt präst i sin hemstad Tübingen, och var god vän till Johannes Kepler. Pascal, som vanligen räknas som räknemaskinens far, konstruerade sin additionsmaskin år 1642, då han var 19 år gammal. Matematikern *Gottfried Wilhelm Leibniz* uppfann en mekanisk multiplikationsmaskin omkring 1670. Tyvärr hade dessa maskiner inte så stor spridning, men de var i alla fall föregångare till de mekaniska räknemaskiner, som uppfanns på 1800–talet (bl.a. Baldwin 1875, Odhner 1878).

Föregångaren till vårs tids datorer anses vara Babbages differensmaskin, uppfunnen av den engelska matematikern *Charles Babbage* (1792-1871). Orsaken till att Babbage började planera en dylik maskin var opålitligheten i den tidens matematiska tabeller. Felen i de gamla tabellerna kopierades ofta över till nya, och många unga matematiker blev först kända genom att de lyckades hitta fel i tabellerna. Det berättas att Babbage beklagade sig för astronomen Herschel och sade att han skulle önska att räkningarna hade blivit utförda med ånga, vartill Herschel svarade, att det vore nog fullt möjligt.

Babbage menade, att man med hjälp av en mekanisk maskin skulle kunna undvika alla fel i de matematiska tabellerna. Maskinen skulle inte bara kunna eliminera räknefelen, utan även alla sättningsfel genom att räkningarna kunde sparas på pappersremsor eller metallskivor, som direkt kunde användas för att producera den tryckta texten.

År 1822 byggde Babbage en prototyp av "differensmaskinen", som han kallade den, eftersom den baserade sig på *differenskalkyl*. Denna metod lämpar sig speciellt väl för att konstruera tabeller. Om man tex vill konstruera en tabell över kuberna av ett antal tal, och man känner början av tabellen, konstruerar man först kubernas differenser, och därpå differensernas differenser (differenserna av andra ordningen). Tredje differenserna är alla 6 (eftersom tredje derivatan av x^3 är 6), vilket betyder att tabellens fortsättning kan konstrueras genom att vända om proceduren: tredje differensen adderas till andra differensen, resultatet adderas till första differensen, varpå det nya resultatet adderas till första differensen och man får en ny kub genom ytterligare en addition. Potensering ersätts alltså med additioner! Metoden kan lätt utvidgas till polynom, vilket gör det möjligt att beräkna också tabeller över funktionsvärden (via polynomapproximationer) med Babbages differensmaskin. Som ett exempel kan nämnas att Babbages modell nr. 2 kan tabulera sjunde gradens polynom med 31 siffrors noggrannhet!

Trots att den var så komplicerad, kunde differensmaskinen inte utföra mer än en uppgift. År 1834 fick Babbage idén till en mer komplicerad maskin, som skulle kunna programmeras i likhet med våra moderna datorer. Den skulle också kunna utföra alla de fyra räknesätten i vilken ordning som helst, och maskinen innehöll också ett ”lager” och en ”kvarn”, som påminner om minnet och processorn i våra dagars datorer. Denna *analytiska maskin*, som han kallade den, kunde programmeras med hålkort. Maskinen skulle kunna läsa in tal med 50 siffror och skriva ut resultaten med 100 siffror. Denna maskin blev dock aldrig byggd, med undantag av en del av kvarnen.

Att Babbage framhärdade i sina försök att planera den analytiska maskinen, antas framförallt bero på den inspiration han fick av Lord Byron's dotter, grevinnan av Lovelace, *Ada Augusta*. Hon hade studerat matematik under ledning av matematikern Augustus de Morgan (känd jämte George Boole som skapare av den symboliska logiken). År 1842 fann hon en artikel om den analytiska maskinen, som skrivits av den italienska ingenjören L.F. Menabrea på franska, och översatte den till engelska. På uppmaning av Babbage bifogade hon sina egna kommentarer till översättningen, som därigenom blev dubbelt så lång som originalet. För att demonstrera principen för den analytiska maskinen, tillade hon också några program. Huruvida det var hon eller Babbage som skrivit dem, är något oklart.

En fungerande version av Babbages differensmaskin nr. 2 blev färdig i England till 200-årsjubiléet av Babbages födelse och kan beses i Science Museum i London. Den visar att Babbages idéer fungerar, även om han saknade de tekniska förutsättningarna att förverkliga dem.

Det följande steget i datorns utveckling var hålkortsmaskinen, som uppfanns av den amerikanske ingenjören *Herman Hollerith* år 1885 för att underlätta den amerikanska folkräkningen. Härigenom kunde data för 62 miljoner amerikaner år 1890 behandlas på en tredjedel av den tid som krävdes år 1880 för 50 miljoner amerikaner. Hollerith's hålkortsmaskiner blev snabbt mycket populära, och år 1911 bildades ett bolag under namnet Computing-Tabulating-Recording Company (C.T.R.), varav senare uppstod IBM (International Business Machines).

Hålkortsmaskinerna kom också till användning vid vetenskapliga beräkningar. 1929 tog *L.J. Comrie*, som var superintendent vid nautiska almanackans kontor i Greenwich, i bruk hålkortsmaskiner vid uppgörandet av Brown's tabeller för månens rörelse. Comrie startade 1937 också ett eget företag, Scientific Computing Services, som blev ledande på sitt område. Liknande beräkningar gjordes också av astronomen Wallace Eckert i USA på 1930–talet. Hans räknebyrå ägnade sig åt tabulering av många slags funktioner, harmonisk analys och integration av differentialekvationer.

En annan typ av maskiner som används för att underlätta beräkningar, är analoga. De ger vanligen lösningen i grafisk form, såsom t.ex. planimetrar och olika astronomiska instrument. På 1870–talet kom lord Kelvin på en idé att använda en analogmaskin för att förutsäga tidvattnet, vilket hade stor betydelse i England. Han insåg att man kunde uttrycka vattnets höjd med hjälp av Fourieranalys som en serie av sinusfunktioner. De olika komponenterna kunde då adderas med en differentialmekanism (*harmonisk syntetisator*), som uppritade vattnets höjd som funktion av tiden på en pappersrulle.

Amplituder och faser för de enskilda komponenterna kunde bestämmas genom att beräkna integralen $\int h(t)s(t)dt$, där $h(t)$ anger tidvattnets höjd och $s(t)$ är sinusvågen för komponenten. Dessa integraler var besvärliga att beräkna för hand, så han använde en mekanisk integrator, som uppfunnits av Lord Kelvins bror James.

Kelvins idéer upptogs på nytt av *Vannevar Bush* vid MIT i USA på 1920–talet, som konstruerade en maskin för att integrera produkten av två funktioner. Integralen kunde sedan ritas upp som en kontinuerlig kurva. Bush insåg också (liksom Kelvin på sin tid), att man också kunde använda sig av återkopppling för att lösa differentialekvationer ekvivalenta med $f(x) = \int f(x)g(x)dx$. En förbättrad version av maskinen kallades differentialanalysator, och blev färdig i början av 1930–talet. Senare versioner av analysatorn använde också elektriska kretsar.

Douglas Hartree som var en teoretisk fysiker i Manchester, England (senare i Cambridge), fick höra om differentialanalysatorn, och beslöt bygga en egen, som han färdigställde år 1934 för en kostnad av 20 pund.

Den första automatiska räknemaskinen konstruerades 1937 vid Harvard universitetet i U.S.A. av *Howard Aiken* i samarbete med IBM. Aiken hade för sin doktorsavhandling i fysik varit tvungen att för hand lösa tusentals olinjära ekvationer, och han tyckte att det borde finnas ett bättre sätt att utföra dessa räkningar. Resultatet blev en 22-sidig rapport som innehöll ett förslag att bygga en automatisk räknemaskin.

Den kallades "Automatic Sequence Controlled Calculator" (A.S.C.C., senare även kallad "Mark I") och bestod av 78 additionsmaskiner, baserade på elektromagnetiska reläer, som kontrollerades av instruktioner på en perforerad pappersrulle av samma typ som användes i självspelande pianon. Den utnyttjade också många av Babbages idéer. Under hela 1940–talet användes den flitigt av amerikanska flottan.

Redan 1915 hade *James Bryce*, en av C.T.R.:s konsulter, föreslagit att man skulle kunna använda elektronrör i databehandlingen, men den första prototypen till en elektronisk dator byggdes först 1938 av *John Atanasoff* och *Clifford Berry* ("ABC"-maskinen). Den kunde utföra binär addition och subtraktion med hjälp av 14 trioder. Två trummor med 1500 kondensatorer bildade minnet ("datorabakus"), och datakommunikationen sköttes med hålkort. Maskinen kunde lösa ett ekvationssystem med 29 obekanta.

Det verkliga genombrottet kom år 1946, då den första egentliga datorn, "Electronic Numerical Integrator and Computer" (ENIAC) togs i bruk. Den hade byggts i University of Pennsylvania av *John Presper Eckert* och *John W. Mauchly* (inspirerad av Atanasoff) med understöd av den amerikanska armén.

Den innehöll 18800 elektronrör, och det räckte två och ett halvt år att löda ihop alla de 500000 ledningarna i datorn. Varje dag brann några av elektronrören, på grund av den stora värmeutvecklingen (150 kW). Den kunde göra 5000 additioner i sekunden, men problemet var att varje gång man ville ändra programmet, måste datorn kopplas på nytt. Minnet rymde endast 20 stycken tiosiffriga tal.

ENIAC var 500 gånger snabbare än de elektromagnetiska datorerna, så en räkning som hade räckt 10 timmar på Mark I nu kunde utföras på en minut! Mauchly och Eckert byggde något senare en ny förbättrad maskin, som fick namnet UNIVAC.

Problemet med elektronrören försvann, då den strömsnåla transistorn uppfanns. År 1961 uppfanns den integrerade kretsen, som kan innehålla tusentals transistorer. Detta revolutionerade ytterligare datortekniken, och gjorde det möjligt att tillverka mycket små datorer, vilket ledde till uppkomsten av mikrodatorerna omkring 1975.

Den snabba utvecklingen inom datorbranschen ledde till grundandet av många datorföretag. Som ett exempel kan vi nämna Digital Equipment Corporation (DEC), som startades av två ingenjörer 1957 i ett gammalt yllespinneri i Maynard, Massachusetts och växte upp till ett av världens ledande datorföretag, som senare övertogs av Compaq, resp. Hewlett Packard. Deras första dator hette helt anspråkslöst PDP-1 (”Programmed Data Processor 1”), som byggdes 1960 och blev känd framförallt genom att den kostade bara 120000 dollar vid en tid då de flesta datorer kostade minst en miljon dollar. Femtiotre av dessa datorer såldes, de flesta till universitet. PDP-1 hade den första grafiska terminalen, och på den skrevs också det första datorspelet, `SPACEWAR`. PDP-8 var den första minidatorn, som massproducerades (kostade endast 18500 dollar) och blev en stor framgång, eftersom den rymdes på ett bord. Ur PDP-11 utvecklades på 1970–talet VAX-serien (”Virtual Addressing Extension”) med 32 bitars ordlängd.

1.2. Räknemetodernas historia

Många av de numeriska metoder, som alltjämt används i beräkningsvetenskapen, har en gammal historia. Metoderna har förstås alltid varit beroende av tillgängliga hjälpmedel. I början fanns det ju inga maskiner, och alla räkningar gjordes för hand, vilket som vi vet, lätt leder till misstag, även för de mest samvetsgranna räknare. Detta insåg också Napier och Briggs när de konstruerade sina tabeller.

Briggs märkte t.ex., att $log(1+x)$ är proportionellt mot x för små x, och att relativt få värden måste beräknas, eftersom andra värden kunde erhållas genom att summera kända logaritmer eller interpolera redan beräknade värden. Detta ledde till utvecklandet av praktiska metoder, som underlättade beräkningarna ofantligt. Differenskalkylen, som all interpolation baserar sig på utvecklades vidare av Newton, Euler och Gauss, för att nämna några av de stora namnen.

Newtons metod, som numera är av mycket stor betydelse vid olinjär optimering eftersom den går lätt att programmera på en dator, kom till användning redan i hans *Principia Mathematica*, men användes troligen av honom redan förut. I sitt arbete om oändliga ekvationer (1669) studerade han ekvationen $y^3 - 2y - 5 = 0$, som har en rot nära 2. Han satte då $y = 2 + p$ och fick den nya ekvationen $p^3 + 6p^2 + 10p - 1 = 0$. Genom att medta endast första ordningens termer fick han $p \approx 0.1$.

Därpå satte han $p = 0.1 + q$ och fick den nya ekvationen $q^3 + 6.3q^2 + 11.23q + 0.061 = 0$. På samma sätt som tidigare fick han $q \approx -0.0054$. Efter ännu ett steg fick han $r \approx 0.00004853$ och $y = 2 + p + q + r \approx 2.09455147$. Vi ser att ekvationen $10p - 1 = 0$ innebär $p = x^2 - x_1 = -f(x_1)/f'(x_1)$, då $f(x) = x^3 - 2x - 5$ och $x_1 = 2$, som stämmer med det moderna sättet att beskriva metoden.

Euler, som var den mest produktiva matematikern under 1700–talet, gjorde också mycket för att utveckla numeriska metoder. Eulers metod för numerisk lösning av differentialekvationer är välkänd, och hans arbete på att utveckla teorin för månens rörelse (som är komplicerad, eftersom månen är så nära jorden) låg till grund för Browns tabeller, som nämndes tidigare.

Karl Friedrich Gauss var mycket intresserad av numeriska metoder, och var också själv en skicklig beräknare. En av hans insatser, som varit av stor betydelse för naturvetenskaperna, är utvecklandet av den minsta kvadratmetoden Även om Laplace fann en mycket elegant framställning av denna metod, var Gauss' framställning ändå enklare, enligt Goldstine. Med hjälp av minsta kvadratmetoden blev det möjligt att beräkna banan för en småplanet ur endast tre observationer, så att den kunde hittas på nytt om den tappats bort. För att lösa normalekvationerna introducerade Gauss sin elineringsmetod. Gauss intresserade sig också mycket för interpolation och studerade Fourierserier. Vad som är mindre känt, är att han också upptäckte vad vi nu kallar den snabba Fouriertransformationen, som återupptäcktes av Cooley och Tukey år 1965.

En annan matematiker som haft stor betydelse för den numeriska analysen under 1800–talet var *Carl Gustav Jacobi*, som bl.a. studerade metoder att lösa lineära ekvationssystem. Han upptäckte en effektiv metod för iterativ lösning av ekvationssystem som domineras av de diagonala termerna. Jacobi upptäckte också en mycket stabil metod för att diagonalisera en symmetrisk matris medels ortogonala transformationer, som återupptäcktes av Goldstine, Murray och v. Neumann i slutet av 1940–talet.

Många av dessa metoder fick senare stor betydelse i dataåldern, eftersom de gick lätt att programmera. Ett stort problem vållade till en början inversion av matriser. För hand var det inte så svårt att med direkta metoder invertera 5×5 matriser, och 10×10 matriser var inte helt omöjliga. Men man märkte snart att stora matriser var i praktiken omöjliga att invertera med direkta metoder ens med hjälp av datorer, på grund av att man måste göra beräkningarna med större och större precision, då matrisens dimension växte. Lösningen var att att använda Gauss elimineringsmetod för att lösa det motsvarande ekvationssystemet. Detta visades i ett epokgörande arbete av John v. Neumann och H.H. Goldstine (*Numerical inverting of matrices of high order*, Bull. Am. Math. Soc. **53** (1947) 1021). Denna artikel var också grundläggande för den moderna numeriska analysen, eftersom den innehöll en noggrann studie av olika typer av fel, som kan uppstå vid numeriska beräkningar (se nästa kapitel).

Numerisk lösning av differentialekvationer är, som bekant, mycket viktig inom naturvetenskapen, och där gjorde Euler en stor insats, som vi redan nämnt (även om Newton redan i Principia hade löst differentialekvationer approximativt). På 1800–talet utvecklades förbättrade metoder *John Couch Adams* och *F.R. Moulton*, som båda var verksamma inom celest mekanik.

De första flerstegsmetoderna introducerades av *Carl Runge* år 1895, och vidareutvecklades av *Karl Heun* år 1900. Följande år introducerades de generella Runge–Kutta metoderna av *Wilhelm Kutta*. En av de första elektroniska datorerna, ENIAC, programmerades för att lösa differentialekvationer med Heuns metod. Problemet var bara att det var besvärligt att ändra programmet, eftersom maskinen måste kopplas på nytt.

1.3. Programmering av datorer

Innan vi nuförtiden kan använda en dator för att lösa ett problem, måste vi först skriva ett program så att datorn vet vad den skall göra. Ett sådant program innehåller ett antal logiska steg, som beskriver den numeriska metod, eller algoritm, som används vid beräkningen.

Som ett exempel skall vi konstruera en enkel algoritm, som beskriver en partikel med massan m som rör sig längs x–axeln under inflytande av en kraft $f(x)$. Rörelsen beskrivs då av Newtons lag:

$$f = ma = m\frac{dv}{dt},$$

där a och v betecknar partikelns acceleration och hastighet. Om tiden delas upp på mycket små intervall $\Delta t = t_{i+1} - t_i$, så vet vi att partikelns hastighet vid tiden t_i approximativt kan anges som medelhastigheten under tidsintervallet $[t_i, t_{i+1}]$:

$$v_i \approx \frac{x_{i+1} - x_i}{t_{i+1} - t_i} = \frac{x_{i+1} - x_i}{\Delta t}.$$

Den motsvarande accelerationen är då medelaccelerationen under samma intervall:

$$a_i \approx \frac{v_{i+1} - v_i}{t_{i+1} - t_i} = \frac{v_{i+1} - v_i}{\Delta t}.$$

Dessa approximationer gäller om tidsintervallet är kort.

Genom att kombinera dessa ekvationer får man en enkel algoritm för att finna partikelns hastighet v_{i+1} och position x_{i+1} vid tiden t_{i+1} ur deras värden vid tidpunkten t_i:

$$x_{i+1} = x_i + \Delta t v_i,$$
$$v_{i+1} = v_i + \frac{\Delta t}{m} f_i,$$

där $f_i = f(x_i)$. Om vi känner partikelns begynnelsehastighet och position, så kan dess position och hastighet vid ett senare ögonblick beräknas ur denna algoritm, som kallas Eulers metod.

För att programmera en sådan algoritm på en dator, behöver vi ett programmeringsspråk. För många år sedan fanns dit inte så många alternativ att välja mellan. Ursprungligen var, som vi sett, den enda möjligheten att programmera på maskinkod eller assembler. FORTRAN var det första högre programmeringsspråket, som uppfanns av *John Backus* och hans medarbetare redan 1957. På 1960–talet tillkom ALGOL (ALGOrithmic Language), som dock inte fick så stor spridning som man hoppats på, kanske för att IBM satsade på FORTRAN.

C–språket och dess varianter (C++, C#) var till en början inte så effektiva vid numerisk programmering men nuförtiden finns det nästan lika goda programbibliotek både för C och Fortran. Fortran–språket har å andra sidan också förbättrats kraftigt genom utvecklandet av den nya standarden, som också möjliggör objektorienterad programmering. Tao Pang har i den senaste upplagan av *An Introduction to Computational Physics* rekommenderat Java–språket (JVM), ett förslag som än så länge verkar något radikalt.

Förutom dessa programmeringsspråk, som förutsätter användning av en kompilator, som översätter programmet till maskinkod, finns det tolkande programmeringsspråk, t.ex. matematikprogrammen MATLAB och MATHEMATICA, där man inte behöver kompilera programkoden först, utan den kan direkt utföras. Detta gör felsökningen betydligt enklare än vid användning av de "klassiska" programmeringsspråken. De tolkande programmen har dock ofta inte ansetts vara särskilt effektiva när det gäller beräkningar, där komplicerade programslingor genomgås tusentals gånger. Numera är det dock möjligt att anropa C– eller Fortran–program från ett MATLAB–program, vilket gör de numeriska beräkningarna betydligt snabbare.

I grundkursen studerade vi redan ganska ingående hur man skriver ett program. Även om denna fortsättningskurs förutsätter att man kan programmera på något språk (C eller Fortran) och kan använda något av de tolkande programmeringsspråken (MATLAB, Mathematica eller Maple), så kan det vara bra att nämna några riktlinjer för god programmering.

Vetenskaplig programmering är egentligen en konstart, som kräver kunskaper i vetenskap (fysik i vårt fall), matematik och datavetenskap, och har som uppgift att lösa ett vetenskapligt problem. Det är givetvis lättare att använda ett kommersiellt program, eller ett program, som någon kollega har gjort, men förr eller senare blir man tvungen att skriva ett eget program. Vilket språk man då väljer, beror på vad man kan, och vad man har tillgång till.

Om man samarbetar med andra forskare, vill man kanske gärna utbyta data och program. Då är det viktigt att skriva program som

- är enkla och lätta att läsa, så att betydelsen av varje del tydligt kommer fram (Att det var lätt för dig att skriva det, betyder inte att du skall göra det svårt för andra),
- dokumenterar sig själva så man förstår, vad de skall göra,
- är lätta att använda,
- är lätta att ändra om så det kan användas på andra datorer,
- man kan låta andra utveckla vidare,

- räknar rätt (mycket viktigt!).

Att skriva ett läsligt program är också viktigt för att det ökar trovärdigheten av dina resultat. Programmet utgör en viktig dokumentation av ditt vetenskapliga projekt.

Skapande artister följer sina egna regler, men som redan nämndes i grundkursen är det till viktigt att kunna programmera modulärt. Detta innebär, att programmet delas upp på delar (subrutiner), som testas var för sig. Dessutom blir programmet överskådligare på detta sätt.

- Skriv gärna många små rutiner, som gör begränsade uppgifter.
- Låt varje modul ha egna invariabler och utvariabler, som överförs genom argumentlistan.
- Om en subrutin är mycket liten och anropas ofta, kan programmet bli oekonomiskt i drift, och kompilatorn optimerar då källkoden bättre om man kombinerar enheter som ofta anropas.

Vänta så länge som möjligt med att skriva ner programmet, försök istället att först sätta dig in i i problemet och definiera det så bra som möjligt. Försök använda pålitliga och enkla algoritmer. Ett snabbt program är bra endast om det också räknar rätt. Kom ihåg att en algoritm som är bra för ett skalärt system inte behöver fungera bra på ett parallellt system.

Ett program som är enkelt och förståeligt kommer också slutligen att innehålla färre fel. Även om det kan ta längre tid att skriva det, lönar det sig ändå i längden. Det kan också leda till att projektet kan avslutas framgångsrikt istället för att man blir tvungen att överge det i ett tidigt skede.

Programmet skall planeras från *uppifrån och ner*. Det betyder att man först skisserar upp vad algoritmen skall göra, och hela tiden håller helheten i sikte.

- Arrangera programmomenten i den ordning de skall utföras.
- Planera detaljerna av varje moment, och dela upp dem på delmoment.
- Fortsätt att dela upp programmomenten tills du når den lägsta subrutinnivån.

Låt programflödet förbli *linjärt*, och undvik att alla hopp ("go to"). Det är viktigt att programmet är strukturerat, dvs uppbyggt av logiska block.

Kom alltid ihåg att spara en uppdaterad fungerande version av programmet, gör alla förändringar på en kopia! Kommentera och dokumentera källkoden så mycket som möjligt, det gör det lättare för dig när du senare vill förändra programmet. I Fortran är det viktigt att deklarera alla variabler (*Implicit none*).