De Programmatuur-Sectie

De keuze van programmeertaal

(C. van de Wijgaart, Technisch Centrum FSW, Universiteit van Amsterdam, april 1978)

Bij de planning van programmatuur, cursussen in programmeertalen en informatica-onderzoek stuit men regelmatig op het probleem welke programmeertaal zich het beste leent voor het beoogde doel. De keus is uiteraard beperkt tot wat op het ondersteunend rekencentrum beschikbaar is. Aldus vallen de nederlandse (en buitenlandse) universiteiten uiteen in clusters van onderling vergelijkbare apparatuur en programmatuur; de overlap in programmeertalen is vrij gering, en zelden wordt een programma zonder aanpassing van het ene merk (type) computer op het andere gedraaid. Dit probleem van de "portabiliteit" toet zich echter zeker niet in alle gevallen voor, waarin binnen sociale en andere wetenschappen behoefte bestaat aan programmering.

Ter illustratie de Amsterdamse universitaire situatie. De beschikbare computerinstallatie is van het merk Control Data, type Cyber, en is min of meer vergelijkbaar met wat de universiteiten van Groningen en Utrecht ter beschikking hebben. De meest gebruikte programmeertalen zijn Fortran, Algol 60 en Pascal *). In het volgende zal ik mij in hoofdzaak tot deze drie talen én Algol 68 beperken. De reden van Algol 68 zal in de loop van dit artikel duidelijk worden.

In verband met een subsidie-aanvraag voor een te ontwikkelen pakket van statistische en data management geöriënteerde standaard-programmatuur schreef ik onlangs een argumentatie voor de gekozen programmeertaal, die ongeveer op het volgende neerkwam.

---

*) Bij voorbeeld in januari '78 was het aantal aanroepen van Fortran: 18270, Algol 60: 5724, Pascal: 3649, Cobol: 2121, Compass (CDC-assembler): 1613, Simula: 1491, Algol 68: 1388, Lisp: 492, Basic: 421.

- Er is gekozen voor Fortran; waar dit persé noodzakelijk is worden kleine gedeelten in assembly language (Compass) toegevoegd.
- Deze keus is niet gebaseerd op aansluiting bij andere programmatuur, hoewel het nieuwe pakket wél in staat zal zijn om een SPSS-system file te gebruiken.
- De keus van Fortran als belangrijkste taal voor dit pakket is, vanuit de informatica bezien, geenszins ideaal; de te gebruiken datastructuren (met name boomstructuren) zouden bijvoorbeeld veel eerder in de richting van Pascal of Algol 68 doen denken.
- Dat desondanks voor Fortran gekozen is heeft de volgende oorzaken:
  - Er zal belangstelling voor dit pakket ontstaan in binnen- en buitenland, zodat spoedig behoefte ontstaat aan conversie naar andere merken en systemen computers. De machine-afhankelijkheid dient dus tot een minimum beperkt te worden, en als high-level programmeertaal moet een taal gekozen worden die zeer algemeen verbreid is en weinig conversieproblemen geeft. Fortran voldoet aan deze eis, andere talen veel minder of in het geheel niet.
  - Het pakket zal zeer omvangrijk materiaal moeten kunnen verwerken. Vele metingen hebben aangetoond dat de efficiency van de CDC-Fortran compiler zeer veel beter is dan die van de -Algol 60 en -Algol 68 compiler.
    Op CDC zullen met Fortran de executietijden acceptabel zijn; voor andere merken wordt hetzelfde verwacht.
  - Het totale pakket zal lineair meerder malen het totaal beschikbare kerngeheugen van de computer-configuratie beslaan. Voor Fortran, via overlay structuur, is dat geen probleem, maar voor Algol en Pascal is dit vrijwel niet oplosbaar. Bij machines met virtueel geheugen doet zich dit probleem overigens niet voor.
  - Het zeer hinderlijke euvel van Fortran: geen dynamische geheugen-allocatie voor arrays, zal worden opgelost via een assembly-routine.

Mutatis mutandis zou, bij ontwikkeling van een dergelijk pakket op een ander merk computer, waarschijnlijk het zelfde gesteld worden, en de uiteindelijke keus dus Fortran zijn. Dat blijkt bijvoorbeeld wel als men bedenkt in welke taal de meer succesvolle statistische pakketten geschreven zijn. SPSS, BMD, MDS, OSIRIS etc. zijn in Fortran; een pakket als CADA, dat in Basic geschreven is, geeft direct ongelofelijke problemen.
Nu kunnen SPSS etc. ook zeer wel om andere redenen in voornamelijk Fortran geschreven zijn: het zijn amerikaanse pakketten, en Fortran is in de Verenigde Staten heel wat vanzelfsprekender dan in Europa.
(Waarschijnlijk dáárom worden Fortran-compilers zoveel beter ondersteund dan de meeste andere.)
Ik zal in het volgende trachten uiteen te zetten waarom de Fortran-keus mij eigenlijk helemaal niet bevalt. Op dit moment moet ik dus helaas concluderen dat voor dit soort pakketten er nauwelijks een andere keus mogelijk is.

Voor "losse" programma's ligt de zaak al heel anders dan voor pakketten, bijvoorbeeld zal de lengte van de programmatekst als regel geen beperkingen opleggen. Meestal zal ook de geprogrammeerde methode niet direct aan een grote "markt" doen denken, zodat conversie wellicht nooit plaats vindt.
Dan worden eenvoud van programmering en efficiency van het programma belangrijk. CDC-gebruikers zullen om dit soort redenen vaker Pascal dan Fortran kiezen. Waarom geen Algol 60 of Algol 68?
Laat ik voor statistische programma's en bijbehorende (niet àl te gecompliceerde) data management wat dieper op de problemen ingaan. Ik zal mij daarbij nogmaals moeten beperken: de vergelijking tussen programmeertalen (of eigenlijk: de implementaties van programmeertalen) is uiterst gecompliceerd. Om enkele aspecten te noemen:
- de definitie van de taal (als die bestaat),
- de geschiktheid van de taal voor modulaire programmering,
- de onderlinge verschillen bij implementatie,

- de eenvoud van gebruik (d.w.z. de kosten van programmering),
- de efficiency van specifieke implementaties,
- het "level" van de taal (al dan niet "dicht bij de machine"),
- de mate waarin de taal is ingevoerd,
- de geschiktheid voor interactief gebruik,
- de geschiktheid van de taal voor diverse soorten toepassingen,
- de snelheid waarmee men de taal kan leren,
- de leesbaarheid van programmateksten,
- de input/output faciliteiten,
- de beschikbare standaard-voorzieningen in de taal,
- de mogelijkheden van linking met andere talen,

enzovoorts. Ik beperk mij hier dan tot het aspect "data-structuren", omdat dit in de praktijk van de sociaal-wetenschappelijke data verwerking m.i. een van de belangrijkste is. ("Datastructuren vormen het fundament van de informatica". Van der Riet, Informatie, 2/78).

In het volgende wordt nagegaan wat de belangrijkste voorzieningen op het punt van data zijn in de eerder genoemde talen, d.w.z. in de "standaard" van deze talen, n.l.:

Fortran - ANSI
Algol 60 (volgens het Algol 60 Revised Report)
Algol 68 (volgens het Algol 68 Revised Report)
Pascal (volgens het Pascal Report).

a. Primitieven

- *Integer* (geheel getal), *real* (gebroken getal) en *boolean* (logische waarde) bestaan in alle vier talen. (Op computer-representaties wordt hier niet ingegaan). De benamingen van deze data-typen zijn niet geheel gelijk; zo is een boolean in Fortran een LOGICAL.
- Fortran kent *dubbellengte* reals: DOUBLE PRECISION.
- Algol 68 kent *extra lange* (variabel) en *extra korte* integers en reals.
- Characters (tekens) komen in Fortran en Algol 60 niet voor; bij Fortran kan men echter wel iets doen bij in- en uitvoer (A-format), terwijl in beide talen strings (zie verder) van één character mogelijk zijn.

Algol 68 en Pascal zijn in dit opzicht veel moderner, of beter: algemener; het character-type bestaat hierin als afzonderlijk type.

- Alleen Pascal kent het algemene *scalar* type: een zelf te definiëren geordende verzameling namen waarop enkele functies (zoals rangnummer) toepasbaar zijn. (Voorbeeld: DAG is een variabele van het type scalar en kan als waarden aannemen: ZATERDAG en ZONDAG.)
- Alleen Pascal kent het *subrange* type, bijv. integers 0 t/m 9 als subrange van het type integer. Gebruik: als type voor array-indices, als check op toegestane waarden van variabelen, bij de definitie van een verzameling (set, zie verder).

b. Gestructureerde data

(Gestructureerd betekent: opgebouwd uit primitieven).

- Arrays: vectoren, matrices etc., het oudste type gestructureerde data. Uiteraard bestaan arrays in alle vier de talen, maar:

  Fortran : maximaal 3 dimensies, vaste lengte (maar "adjustable" in subroutines).

  Algol 60 : onbeperkt aantal dimensies, variabele lengte ("dynamisch" is het goede woord: tijdens de executie wordt de lengte van de arrays vastgesteld),

  Algol 68 : onbeperkt aantal dimensies, variabele lengte. Ook constante arrays bestaan (denk aan tabellen). In tegenstelling tot Fortran en Algol 60 mogen array elementen niet alleen integer, real en boolean zijn, maar bijv. ook character, string en array.
  Algol 68 kent een beperkte "matrix-notatie" (vectornotatie etc.), kent "slices" (bijv. een rij van een matrix als ineens te gebruiken variabele), en kent "flexible array bounds" (de grenzen van een flexible array mogen tijdens de executie gewijzigd worden).

Pascal : onbeperkt aantal dimensies, maar vaste lengte. Als index-type is in principe de scalar toegestaan (dus niet alleen de integer); als typen van de array elementen is elk ander type (dus ook record en file, zie verder) toegestaan.

- Strings (reeksen tekens)

Fortran : zeer beperkt, n.l. alleen als (assigneerbare) Hollerith-constante, en bij in- en uitvoer (A-format).

Algol 60 : alleen als procedure-parameter, en daardoor alleen zinvol bij uitvoer.

Algol 68 : een string is gedefinieerd als een flexible array van characters; het is dus een zelfstandig type met variabele lengte. Vele string-operaties zijn mogelijk, zoals substring, n-de symbool, inlezen van strings, concatenation.

Pascal : Gedefinieerd als packed (zie verder) array van characters, dus evenals bij Algol 68: een apart type. Via wijziging van dit packed array bestaan beperkte mogelijkheden van stringmanipulatie.

- Files (structuren van één type componenten, die op achtergrondgeheugen zijn opgeslagen en toegankelijk zijn via buffers. Er is dus een read/write statement nodig voor overdracht naar/van een variabele van het component type.)

Fortran en Algol 60: geen files aanwezig.
(Dit is natuurlijk uiterst merkwaardig! Vanzelfsprekend dat iedere compiler-bouwer daar wat aan doet. Bijv. is voor CDC-Fortran hiertoe het "program statement" bedacht, waarin files als parameters toegestaan zijn, die omgezet worden in "device nummers" voor I/O. Evenzo is voor CDC-Algol 60 de channelkaart uitgevonden.
Aangezien de gehele I/O in Algol 60 niet gedefinieerd

is, is het ontbreken van files hiervan het gevolg. Uiteraard werkt e.e.a. zeer negatief op de portabiliteit van programmatuur.)

Algol 68 : Ook voor files biedt Algol 68 weer veel: coded files, binary files, compressible files, en zelfs: random files.

Pascal : De structuur van een file (zowel intern als extern) kan zeer complex zijn, bijv. files van records en van arrays.

- Records. (Een record ') is een gestructureerd data type waarvan de componenten niet van eenzelfde type behoeven te zijn. Bijv. kan een record dus bestaan uit twee integers en een string.)
  Fortran en Algol 60 kennen geen records.
  Algol 68 kent wel records. Een of meer van de componenten kan van het pointer type zijn, waardoor lijsten en netwerken van records gecreëerd kunnen worden.
  Pascal : geheel als bij Algol 68, maar bovendien mag een record een "variant part" hebben (bijv. "mannen" en "vrouwen" hebben gedeeltelijk verschillende structuren, gedeeltelijk dezelfde, maar beide zijn van het record type "persoon").

- Complex (complexe getallen, dus a + bi, met a en b real).
  Fortran : standaard aanwezig.
  Algol 60 : afwezig.
  Algol 68 : gedefinieerd als record met componenten a en b (zie boven). Ook de vereiste operaties zijn standaard.
  Pascal : afwezig, maar via records en procedures zelf programmeerbaar.

- Sets (verzamelingen. Bijv. de verzameling der medeklinkers). Alleen in Pascal aanwezig, met operaties als doorsnede, deelverzameling, element van. De set in Pascal is wel beperkt in omvang (relatie met de bits in een woord). Eenvoudige toepassingen van de set zijn vaak al erg aardig, zoals

---

') Dit is de Pascal-naam; in Algol 68 heet een record een structure. Niet te verwarren met de "records" waaruit een file is opgebouwd.

ALFABETIC := SYMBOL IN ['A'..'Z']

(In Algol 68 kan men sets wel zelf definiëren, vooral omdat deze taal de mogelijkheid biedt zelf operaties te definiëren.)

- Bits.

  Alleen in Algol 68: een variabele van het type bits komt overeen met een machinewoord, opgebouwd uit booleans. De notatie is binair, octaal etc. Diverse operaties zijn mogelijk.

- Bytes.

  Alleen in Algol 68: een bytes-variabele komt overeen met een machinewoord, opgebouwd uit characters. Bytes ontstaan door packing van strings. Ook hier zijn diverse operaties mogelijk.

c. Packed structures

Alleen Pascal heeft algemene pack-voorzieningen: array, record, set en file mogen als packed structure gedefinieerd worden, wat (mogelijke) besparing van geheugenruimte tot gevolg heeft.
Vermeld dient te worden dat het gebruik van packing in Pascal op twee manieren mogelijk is: simpelweg declareren van een packed structure (de rest zoekt de compiler uit; de programmeur gebruikt de packed structure op de zelfde manier als een niet ge-pack-te), maar voor arrays bovendien (en dat gaat sneller): een gewoon array kan via een opdracht gecopieerd worden in een packed array, en omgekeerd.
In Algol 68 wordt pack/unpack gebruikt om strings naar bytes te transformeren, boolean arrays naar bits, en terug.
Voor Algol 68 wordt bovendien gewezen op structuren van verkorte primitieven. (De compressed file is iets anders dan een packed file.)
Algol 60 en Fortran kennen geen packing (al wordt in veel compilers wel iets gedaan, bijv. voor boolean array-elementen 1 bit in plaats van 1 woord).

d. Pointers

Fortran en Algol 60 kennen geen pointers, Algol 68 en Pascal wel.
Een pointer is een variabele, die het adres geeft van een andere variabele. Wanneer pointers in records worden ingebouwd heeft men de mogelijkheid om records het adres mee te geven van andere records,

dus om list processing te gaan doen (boomstructuren etc.).
Door pointers aangewezen variabelen kunnen dynamisch (dus tijdens de executie) gegenereerd worden.
In Algol 68 is de pointer (daar geheten: reference to ...) een zeer essentieel begrip in de gehele taal. Theoretisch zijn de mogelijkheden groter dan bij Pascal (bijv. pointers naar pointers), maar practisch gezien is het lastig dat men zich in Algol 68 voortdurend bewust moet zijn van de MODE van een variabele, van het al dan niet "reference to", van begrippen als dereferencing. Dit maakt de taal moeilijk en veroorzaakt licht fouten (ook als er helemaal geen pointers gebruikt worden).

In het kort:

		Fortran	Algol 60	Algol 68	Pascal
primitieven	integer	++	++	++	++
	real	++	++	++	++
	boolean	++	++	++	++
	character			++	++
	long (short)	+		++	
	scalar				++
	subrange				++
structured	array	+	++	++	+
	string	+	+	++	++
	file			++	++
	record			++	++
	complex	++		++	
	bits			++	
	bytes			++	
	set				++
packing				+	++
pointers				++	++

Legenda: blanco : afwezig
+ : beperkt aanwezig
++ : goed in voorzien.

Uit de gegeven tabel blijkt dat de verschillen aanzienlijk zijn. Men dient zich hierbij te realiseren dat FORTRAN (FORmula TRANslation language) en ALGOL 60 (ALGOrithmic Language 1960) bedoeld waren voor "computation", voor rekenwerk dus, en dat deze talen intussen diverse computergeneraties oud zijn, terwijl Algol 68 en Pascal later ontwikkeld werden en bedoeld zijn als "algemene programmeertaal".

Bij de ontwikkeling van nieuwe programmatuur dient de programmeur (indien hij meer dan één taal ter beschikking heeft) zich van geval tot geval in zijn keuze van programmeertaal te baseeren op een lange rij van aspecten, zoals in het begin van dit artikel gesteld. Vaak geven één of enkele aspecten de doorslag. Zo wordt in veel gevallen Fortran gekozen vanwege portabiliteit en snelheid, en wordt het gebruik van dynamische arrays daaraan opgeofferd. (Ofwel: men wijkt zeer ver van ANSI af en lost dit laatste via assembler op.)

Het onderwerp "data" is maar een van de aspecten die de programmeur bij zijn keuze hanteert. Aan talen als Fortran en Algol 60 is wel te zien dat in het verleden dit aspect bepaald geen doorslag gevende rol heeft gespeeld, maar aan Algol 68 en Pascal dat er sinds ca. 1965 in dit opzicht veel veranderd is.
De universitaire computer is niet langer het monopolie van de klassieke "rekenaar", geheel andere toepassingen zijn sterk in opkomst, en dit geldt zeker voor het gebied van de Sociale Wetenschappen. Denk bijv. aan simulaties, computer assisted instruction, tekst verwerking.

Op dit moment vormt statistische programmatuur, en de daarbij noodzakelijke data management, het overgrote deel van wat binnen Nederland in de Sociale Wetenschappen aan software ontwikkeld wordt. (Ik ga hier met opzet voorbij aan on-line toepassingen, waarvoor geheel andere normen gelden.) De oorzaak is waarschijnlijk dat de opdrachtgevers van statistische programmatuur, de methodologen met name, destijds voldoende "rekenaar" waren om te passen in het gezelschap van matrix-bewerkende Fortran en Algol 60 gebruikers. Sindsdien is tien, vijftien jaar verstreken, waarin rekenaars informatici werden, EL-X1's werden opgevolgd door CDC-Cybers, en talen als Algol 68 ontwikkeld werden.
Maar Fortran bleef.

Uit deze vergelijking van data-structuren valt te concluderen:

- Algol 68 en Pascal zijn, vanuit de geboden data-faciliteiten beschouwd, geheel andere talen dan Fortran en Pascal. Blijkbaar zijn zij bedoeld voor ruimere toepassingen dan de voornamelijk numerieke en statistische van tot voor kort. Letwel::Algol 68 en Pascal bieden óók wat Fortran en Algol 60 hebben, maar daarnaast meer.
- Afgezien van de efficiency van Algol 60 compilers: het ontbreken van in- en uitvoerdefinities in deze taal, en daardoor de grote verschillen in implementaties, maakt Algol 60 zélfs ongeschikt om er nog langer numerieke en statistische toepassingen in te bedrijven. Om van andere toepassingen maar niet te spreken.
- Wanneer momenteel in de Sociale Wetenschappen aan andere dan statistische etc. toepassingen wordt gedacht zal het data-arsenaal van Fortran veelal ernstig tekort schieten; bij voorkeur moet daarom hiervoor aan andere talen gedacht worden.
  Voor data management en statistiek schijnt Fortran vooral gebruikt te worden omdat ánderen Fortran gebruiken. Een slechte reden.
- Algol 68 biedt enorm veel, misschien iets te veel om er niet door afgeschrikt te worden; de taal is niet eenvoudig, zeker niet voor een beginneling, en de complexiteit werkt ongetwijfeld belemmerend op de efficiency. De vraag is overigens voor hoeveel programma's dit belangrijk is. Voor gecompliceerde toepassingen, voor professionals, voor informatici lijkt me dit echter de taal bij uitstek.
- Pascal is duidelijk bezig een grote markt te veroveren, geen wonder, het koppelt de algemeenheid en vele faciliteiten van Algol 68 via o.a. (misschien net iets te veel) beperkingen aan eenvoud. De beginneling krijgt met Pascal in korte tijd toegang tot belangrijk meer dan rekenarij, de gevorderde beschikt in Pascal over een taal met mogelijkheden van snelle programmering, leesbare programma's en zoveel data-mogelijkheden dat het zijn behoefte aan creativiteit zal blijven prikkelen.

Een geheel ander uitgangspunt voor een discussie over programmeertalen vormt de informatica. Ik bedoel hiermee niet dat informatica zich niet met de principes van standaardprogrammering zou bezig houden, of dat informatica te onpractisch zou zijn om efficiency etc. naar waarde te kunnen schatten, integendeel. De informatica zal zich echter óók met

andere problemen dan standaardprogrammatuur bezig houden, bijv. met de ontwikkeling van algoritmes en allerlei andere terreinen van onderzoek, en met onderwijs.
Aan de Universiteit van Amsterdam wordt op dit moment gewerkt aan de totstandkoming van een bijvak "sociaalwetenschappelijk geöriënteerde informatica". In dit kader kan hierbij opgemerkt worden dat programmeertalen meer aandacht zullen krijgen dan in tot op heden gegeven cursussen (meer talen, meer fundamenteel), en dat bij het verrichten van toegepast informatica-onderzoek een geheel ander taalgebruik kan ontstaan dan tot dusver. (De andere aspecten van dit bijvak zijn hier niet aan de orde.)

Onderwijs programmeertalen

Ik ga hier niet uitgebreid op dit onderwerp in, en beperk mij zelfs tot de eerste programmeertaal die men studenten sociale wetenschappen zou moeten leren. Die eerste taal is overigens veelal bepalend voor iemands verdere activiteiten in de informatica.
Eisen, waaraan deze taal volgens mij zou moeten voldoen, zijn:
- behoorlijk gedefinieerd, ook wat invoer/uitvoer betreft (dus niet: Fortran en Algol 60),
- modern, in de zin van datastructuren en syntax (al weer vallen Fortran en Algol 60 af),
- betrekkelijk eenvoudig (nu wordt het moeilijk voor Algol 68).

Weinig relevante eisen zouden hier zijn: portabiliteit, efficiency, etc.
De uitkomst is duidelijk: Pascal.

Programmeertalen in onderzoek

Behalve op enkele terreinen als real time toepassingen, kunstmatige intelligentie en statistische verwerking wordt op dit moment nog niet veel aan sociaal-wetenschappelijke informatica gedaan. Te gebruiken talen zullen in hoge mate bepaald worden door het onderzoeks-onderwerp. In veel gevallen zal men bij een weinig algemene taal uitkomen. Enkele voorbeelden hiervan:
- Op dit moment wordt geëxperimenteerd met het computer based instruction systeem PLATO. "De" programmeertaal hiervoor is : TUTOR.
- Voor tekstverwerking bestaan talen als SNOBOL en LISP. Men zou echter ook met Algol 68 een heleboel kunnen doen.

- Ook voor simulatieproblemen zijn aparte talen ontwikkeld. Zeer de moeite waard is SIMULA, dat Algol 60 als subset heeft. (De I/O wijkt sterk af van wat meestal in Algol 60 compilers voorkomt.) Dat betekent bijvoorbeeld dat bestaande Algol 60-bibliotheken van numerieke enz. procedures in SIMULA bruikbaar zijn. Ook hier geldt weer: ook Algol 68 biedt mogelijkheden.

Wat men zich hierbij ongetwijfeld zal afvragen is o.a.: hoe theoretisch of practisch wordt dit soort onderzoek? Wordt concrete programmatuur ooit product van dergelijk onderzoek, of wordt in eerste instantie gezocht naar structuren, algoritmes, oplossingsmethoden? Speelt uiteindelijk efficiency een rol? Uitwisselbaarheid? De antwoorden zijn nog moeilijk te geven.
Ik zou er zeker wél een voorstander van zijn dat bij publicatie van onderzoeksresultaten de voorkeur gegeven wordt aan een machine-onafhankelijke "taal" (desnoods een zónder implementatie), die in het algemeen begrijpelijk is voor vakgenoten, voor beroepsprogrammeurs en voor sociale wetenschappers die hiermee te maken hebben. In veel moderne informatica-literatuur wordt (ongetwijfeld om dit soort redenen) gekozen voor Pascal of Algol 68. Gezien de "algemeenheid" van deze talen, d.w.z. de geschiktheid van deze talen voor uiteenlopende toepassingen, lijkt mij daarom dat ernstig overwogen zou moeten worden in principe Pascal of Algol 68 te gebruiken.

Voor één toepassing, n.l. tekstverwerking, wil ik hier ten slotte nog wat verder op ingaan, waarbij ik voorbij ga aan speciale talen als SNOBOL en LISP, en mij beperk tot Pascal en Algol 68.

Pascal biedt, in het kort, voor tekstverwerking de volgende faciliteiten.
- De "character" is een afzonderlijk type, als constante en als variabele te gebruiken. De string bestaat als constante. Voor een variabele string gebruikt men een packed array (zie hiervoor) van characters, en omdat array bounds in Pascal vast zijn heeft een string dus een vaste lengte.
- De (on)gelijkheidsoperaties zijn gedefinieerd, zowel voor characters als voor strings, afhankelijk van de locale character set.
- Wijziging van string-componenten is charactergewijs mogelijk, enwel zonder de pack-instructies te gebruiken.

- Standaardfuncties zijn: volgende character (van de set), vorige character, rangnummer van de character in de set, en inverse daarvan.
- Voorbeelden van gebruik van character en string in andere types: files van char/strings, arrays van char/strings, char/strings als componenten van records (dus ook in lijsten), subrange en set van characters.
- Invoer: charactergewijs.
  Uitvoer: charactergewijs en string ineens.
  Er zijn weinig aanvullende I/O procedures.

Een eenvoudig voorbeeld: het inlezen van een woord in een tekst.
We veronderstellen dat de eerste letter al gelezen is, zodat we weten dat er nu inderdaad een woord komt.

```
K:=0;
REPEAT K:=K+1; WOORD[K]:=CH; READ(CH)
UNTIL (NOT(CH IN ['A'..'Z']) OR (K=LENGTE) OR EOF)
```

wat betekent:

- stel K op nul,
- herhaal: verhoog K met 1, maak de K-de letter van "WOORD" gelijk aan de gelezen character, lees de volgende character,
- totdat: de gelezen character niet tot de verzameling A t/m Z behoort, of de (vaste) stringlengte overschreden dreigt te worden, of er een end of file volgt.

Vergeleken bij veel andere talen ziet dit er niet onredelijk uit. We zullen straks zien hoe het in Algol 68 kan.

De verschillen van Algol 68 met Pascal inzake faciliteiten voor tekstverwerking zijn, heel in 't kort:

- De string bestaat óók als variabele, en is dan bovendien variabel in lengte. In feite zijn strings arrays van characters met flexibele grenzen. Strings kunnen gepacked worden tot bytes.
- Er zijn belangrijke andere operatoren:
  Aaneenschakeling: "AB" + "CDE" levert: "ABCDE"
  vermenigvuldiging: 3 * "AB" levert: "ABABAB"
  stringlengte: upb "ABC" levert: 3.
- Zonder over arrays te praten kan men werken met
  het n-de character: als S = "NAME" dan S[upb S] = "E"
  de substring: bij de zelfde string S: S[2:3] = "AM"

- Vooral bij I/O zijn er onder de standaard procedures enkele belangrijke, de "event procedures".
  Bijvoorbeeld de procedure ON LOGICAL FILE END wordt éénmaal in het programma aangeroepen, en dan verder automatisch zodra dat zin heeft. De programmeur schrijft dus éénmaal wat er moet gebeuren zodra ergens een logical file end bereikt wordt; bij Pascal moet dat bij elke leesopdracht.
  Idem: de procedure MAKE TERM zorgt ervoor dat het lezen van strings automatisch beëindigd wordt wanneer een symbool uit een opgegeven string wordt ingelezen.
- O.a. biedt Algol 68 random access files.
- Er zijn zeer uitgebreide voorzieningen voor formatted I/O.

Terug naar het lezen van een woord. In het overeenkomstig Algol 68 programma wordt tevoren MAKE TERM en ON LOGICAL FILE END aangeroepen, dat hoeft dus nu niet meer. We gaan van hetzelfde startpunt uit als bij het Pascal programma, d.w.z. de eerste letter van het woord is gevonden. Het vereiste programma is dan:

```
READ(BACKSPACE); READ(WOORD)
```

d.i.: zorg dat de eerste letter óók in het WOORD belandt, d.w.z. lees een character terug, en lees dan WOORD in, totdat MAKE TERM of ON LOGICAL FILE END werkt.

Eerlijkheidshalve vermeld ik hierbij dat het Algol 68 programma nogal wat trager is (op CDC) dan het Pascal programma. Het is de vraag of dat hier belangrijk is.
Het verschil met het Pascal programma kan misschien als volgt worden uitgedrukt: Pascal spelt woorden, Algol 68 leest woorden ineens, die bovendien onbeperkt lang mogen zijn. Bij kinderen in het basisonderwijs vinden we dit een héél verschil in ontwikkeling.

Literatuur

Control Data Corporation: Fortran Extended Reference Manual version 4 (waarin aangegeven: ANSI standards)

K.Jensen & N.Wirth: Pascal, User Manual and Report. Springer Verlag, Berlijn, 1975.

P.Naur (Ed.): Revised Report on the Algorithmic Language Algol 60. Reprint: Regnecentralen, Kopenhagen, 1964.

F.G.Pagan: A practical Guide to Algol 68. Wiley, New York, 1976.

C.v.d.Wijgaart: Inleiding programmeren in Algol 60. Syllabus, Technisch Centrum FSW, Amsterdam, 1972.

C.v.d.Wijgaart: Inleiding programmeren in Pascal. Academic Service, Den Haag, 1977.

A.v.Wijngaarden e.a.: Revised Report on the Algorithmic Language Algol 68. Mathematisch Centrum, Amsterdam, 1975.