Over standaardroutines.

Voor een machine met een trommel is het, naar men zich hopenliji denken kan, een belangrijk criterium, waarnaar men standaardroutines kan onderverdelen, of zij permanent op een vaste plaats in de kernen aanwezig geacht kunnen worden of niet.

Onder de permanent aanwezigen vallen in elk geval de zg. coordinatorroutines, zoals de aanroepen van de P- en V-operaties. In de ALGOL-machine zullen ook de systeemroutines hierbij horen, in het bijzonder die stukken programma, diete maken hebben met de aan- of afwezigheid van informatie in de kernen. (Je kunt uit de aard der zaak informatie "facultatief" op de kernen hebben: als het programma, dat deze aanwezigheid onderzoekt, ook zelf afwezig kan zijn, dan krijg je kennelijk rare dingen.) Het is duidelijk, dat wij de totale omvang van deze routines vanwege hun permanente aanwezigheid scherp in het oog moeten houden.

Het ALGOL-systeem zal de beschikking krijgen over een uitbreidbare bibliotheek, nl. van standaardprocedures; ik neem aan, dat deze bibliotheek zich permanent op de trommel zal bevinden. Bij de incorporering en de interne formulering treedt direct een moeilijkheid op.

Men kan de bibliotheek op de trommel beschouwen als "moedercopie" en bij vertaling van een bibliotheek gebruikend ALGOL-programma de gebruikte bibliotheekprocedures in het objectprogramma opnemen. Deze opname wordt geen blindelingse copiering; nomenclatuur (identificatie van geheugenbezettingen, andere bibliotheekprocedures) moet dan vertaald worden. De copie bevat immers ook de informatie hoe hij -en andere bibliotheekprogramma's- in het ALGOL-programma zijn ingepast. Op deze wijze is de bibliotheek louter een vertalerkwestie, waar het run time system nauwelijks iets mee te ~~MK~~ maken heeft: in het objectprogramma hoeft niet meer te zien te zijn, dat sommige procedures niet expliciet gedeclareerd zijn geweest.

Dit is evenwel niet de weg, die wij in wilden slaan. Ten eerste waren we- terecbt of niet- een beetje bang voor de complicatie van de vertaler; ten tweede betekent dit, dat als twee ALGOL-programma's dezelfde standaaddprocedure gebruiken, dat deze procedure dan inmiddels drie maal in het geheugen staat en mogelijk tweemaal op de kernen. En de vraag is, waarom zouden we dat doen?

We mikken dus nu op een oplossing, die de bibliotheekroutines gebruikt zoals ze op de trommel staan. Wat we winnen is een simplificaate van de vertaler en de winst, dat we zuiniger met het geheugen omspringen. Mogelijk winnen we ook, dat de tekst van de bibliotheekprocedures, die nu immers niet meer door de vertaler vermalen worden, met wat grotere vrijheden opgesteld kan worden.

De prijs, die we betalen, is vooralsnog onbekend. Ik zie op het ogenblik twee betalingen.

Doordat verschillende programma's dezelfde kernpagina nu parallel kunnen willen gebruiken, wordt het concept van een "heilige kernpagina" gecompliceerder. We kunnen niet meer met een enkel bit volstaan , maar hebben een teller nodig, die bij heiliging met 1 verhoogd moet worden en bij profanatie ~~M~~ met 1 verlaagd moet worden. Alleen als de teller = 0 is, is de pagina vogelvrij.

De tweede prijs is een kerngeheugenbezetting door een "bibliotheeksinhoudsopgave", die bijhoudt, waar de bibliotheekpagina's zich bevinden. De lengte van deze tabel is evenredig met de omvang van de bibliotheek, of de bibliotheek nu gebruikt wordt of niet. Dit ~~MX~~ is strict genomen natuurlijk tegen de regels, maar omdat deze tabel waarschijnlijk maar 1 woord zal beslaan per trommelpagina van waarscijnlijk 512 woorden, dabhten we, dat we ons dit konden permitteren. ALGOL-programma's identificeren bibliotheekprocedures -en bibliotheekprocedures identificeren zichzelf en elkaar- invariant in termen van deze tabel.

Als dit normale procedures zijn, dan krijgen ze bij activering hun geheugenruimte toegewezen in de stapel van het activerende programma. Dit is mogelijk en geeft geen aanleiding tot extra complicaties, bij gratie van het feit, dat de activiteit van een dergelijke bibliotheekprocedure synchroon verloopt met het aanroepende programma. Van de aanroep van een bibliotheekprocedure hoeft, voorzover ik zie, de coordinator geen bijzondere notitie te nemen.

Het bovenstaande is slechts ter inleiding, ter verhoging van de contrastwerking met wat volgt. Wij zullen nl. de individuele ALGOL-programma's ook de beschikking moeten kunnen geven over geprogrammeerde standaardhandelingen, die asynchroon met het ALGOL-programma moeten kunnen werken. Ik voorzie nu, dat het een drommels verschil zal maken, of een dergelijk asynchroon proces willekeurig veelvoudig parallel geactiveerd moet kunnen worden of niet. Als wij ons ervan konden overtuigen, dat dit niet het geval zal zijn, dan zou ik van ernstige zorgen bevrijd zijn.

Straks zal ik, ter illustratie, als intellectuele exercitie en mogelijk voor later gebruik een voorbeeld proberen uit te werken, dat niet parallel geactiveerd hoeft te worden. Eerst wil ik proberen aan te duiden, waar ik de moeilijkheden verwacht, als aan deze voorwaarde niet voldaan is.

Als wij even aannemen, dat elk werkend ALGOL-programma het samenspel impliceert tussen dit programma en een aan dit ALGOL-programma toegevoegd simultaan werkend proces, een soort prive-secretaresse, dan betekent de introductie van een ALGOL-programma de creatie van twee abstracte machines, te weten het ALGOL-programma en de prive-secretaresse. Ik laat nu even in het midden of de programmatekst van de prive-secretaresses in veelvoud of in enkelvoud aanwezig zal zijn. In het ene geval moet ik een multipliceringsmechanisme maken, in het andere geval een extra activeringsmechanisme. Wat mij zorgen baart is, dat er tussen ALGOL-programma en prive-secretaresse informatieverkeer zal moeten plaatsvinden, dat er seinpalen zullen moeten zijn, die voor beiden toegankelijk zullen moeten zijn. De introductie van een ALGOL-programma betekent dan ook niet "alleen maar" de creatie van twee nieuwe abstracte machines, het betekent de creatie van twee op bijzondere wijze op elkaar afgestemde abstracte machines.Ik kan bijvoorbeeld niet overzien, of we ons zullen kunnen permitteren, om deze seinpalen gedurendehun leven op vaste geheugenplaatsen onder te brengen. Zo nee, dan bekruipt mij plotseling de angst, dat "verschuiven" van die informatie een hel wordt, omdat meer dan 1 abstracte machine daar weet van moet hebben. Kortom, "wat en waar" is de common ground waar de twee elkaar kunnen ontmoeten?

Opm. Voor de bibliotheekprocedures hebben we de "common ground" geschapen door de tijdens run time permanent aanwezige "bibliotheekinhoudsopgave"; tijdens vertalen komt een soort complement van deze tabel ter sprake, nl. als "prevulling van de naamlijst": aan de standaardnamen zijn daar dan verwijzingen toegevoegd naar de bibliotheekinhoudsopgave. Wijziging van de bibliotheek impliceert in het algemeen een herindeling van het totale bibliotheekgeheugen en is geen triviale opgave. Wij zijn er essentieel van uitgegaan, dat hergroepering van de bibliotheek zal gebeuren tijdens executie, of tussen vertaald zijn en uitvoering, van een of meer ALGOL-programma's. Ziehier een nieuw voordeel van load and go!

Deze zorgen zijn allemaal aanmerkelijk minder, wanneer we een asynchrone "manus voor allen" slechts in enkelvoud willen kunnen bespelen. Stel, dat een ALGOL-programma mogelijkerwijze, maar niet noodzakelijkerwijze, deze manus wil kunnen gebruiken. (Ik ga straks een en ander illustreren aan de printer; een programma mag printen, maar hoeft dat natuurlijk niet!). Van deze manus maak je dan, eens en vooral, een stamgast van je hotel voor abstracte machines. Als hij klein is, zet je hem op vaste plaatsen op de kernen, anders laat je hem als ieder ander programma via de paginaadministratie meedraaien. In elk geval kan je je veroorloven om hem permanent een klein beetje levend geheugen permanent ter beschikking te stellen voor het informatieverkeer.

(De opmerking is, dat als geen van de lopende ALGOL-programma's van de printer gebruik wil maken, er dus waarschijnlijk niet zoveel te doen is en dat je je deze kleine vaste bezetting dus wel kunt permitteren. En er hoeft er maar eentje te willen printen en je hebt het er al uit.) Op dit moment is er een ondubbelzinnige, invariante terminologie, waarin de onderscheiden ALGOL-programma's de manus kunnen toespreken.

Omdat het woord "pagina" al gebruikt is voor de onderverdeling van kern - en trommelgeheugen, noem ik een pagina, zoals die door de printer bedrukt kan worden, een formulier, dwz. het stuk van perforatie tot perforatie.

Ik beschouw een de printer als een snel uitvoerorgaan, zo snel, dat

a) ALGOL-programma's hun resultaten zo ongeveer synchroon met de productie op papier kunnen krijgen, en sterker
b) de printer zelfs meer dan 1 programma bedienen kan.

Het idee is, om de printer steeds formuliersgewijze aan een bepaald programma te gunnen. Dit impliceert, dat bovenaan elk formulier ongevraagd geprint wordt een identificatie van het producerende programma, een volgnummer en verder wat nodig geacht wordt, datum etc.. De "expeditie" moet nu de formulieren maar afscheuren en sorteren.

Dit impliceert, dat een ALGOL-programma pas de printer maag aanvragen, wanneer het de gegevens voor een volledig formulier klaar heeft. Immers: zou een ALGOL-programma de eerste helft van een formulier kunnen bedrukken, voordat de resultaten voor de tweede helft geproduceerd zijn, dan zou dit programma de printer heel lang kunnen blokkeren, omdat de productie van de volgende resultaten wel eens heel lang op zich zou kunnen laten wachten. Elk ALGOL-programma, dat van de printer gebruik maakt, heeft dus een prive-buffer met de capaciteit van een formulier. Ald we aannemen, dat op een formulier 64 regels geprint worden, dan komen we met 32 woorden per regel op een buffer van 2048 woorden, dwz. met een paginagrootte van 512 woorden dus 4 pagina's.

Naast de privebuffer heeft elk ALGOL-programma als globale grootheden de administratieve gegevens over de staat van vulling van de buffer, formuliernummer, etc. Elk ALGOL-programma begint met deze gegevens passend te initialiseren als standaard onderdeel van START.

Een ALGOL-programma, dat wil printen bouwt nu, helemaal locaal, een volledig formulierbeeld op. Hierbij maakt het, neem ik aan, gebruik van in enkelvoud aanwezige, conversie en opmaakroutines. Het is hier onbelangrijk of deze permanent op de kernen staan,dan wel als bibliotheekprocedures van de trommel gehaald kunnen moeten worden. Belangrijk is, dat deze operaties synchroon met het werkende programma plaatsvinden. Ik neem aan, dat deze "conversie en opmaakroutines" detecteert, wanneer aan een nieuw formulier begonnen wordt en op grond daarvan de formulierkop invult en tevens detecteert, wanneer een formulier vol is.

Zodra een formulier vol is, moet het programma met de buitenwereld gaan communiceren, dwz. met de formulierprinter. Deze formulierprinter is de bovengenoemde, geprogrammeerde "manus voor allen".

De synchronisatie met de formulierprinter loopt over twee seinpalen; dit zijn twee vaste geheugenplaatsen, waarvan ALGOL-programma en formulierprinter beide kennis moeten dragen. Verder is er de "toonbank", waarover het formulierbeeld wordt aangeboden. Als we de informatie lijfelijk zouden transporteren, zou dit een stuk van 2048 plaatsen zijn.

Als we een formulierbuffer beschouwen als een groot array van vier pagina's, dan hoeft aanbod helemaal niet het lijfelijke transport van 2048 woorden te impliceren. Immers: die vier pagina's zijn bereikbaar via vier descriptoren, de zg. TPV's (Trommel Pagina Variabelen). Als we deze vier TPV's transporteren, bereiken we daarmee, dat de vier gevulde pagina's, die eerst tot het producerende ALGOL-programma behoorden, nu zijn overgeheveld naar de formulierprinter.

De organisatie van de grote arrays zal de feitelijke reservering van Pagina's niet doen bij de declaratie van het array -dan worden alleen de TPV's geintroduceerd en geinitialiseerd- maar pas bij werkelijk gebruik van het eerste element van de pagina. Na beeindiging van interesse -zoals bij blokverlating- worden detrommelpagina's weer aan de vrije hoop toegevoegd

Een en ander heeft tot gevolg, dat wij zonder absurde onkosten de individuele ALGOL-programma's en de formulierprinter grotere buffers zouden kunnen geven; als wij dat niet doen, zullen wij dus iets van een motivering moeten geven.

Een reden om de individuele programma's buffers van meer dan 1 formulier te geven zou de wens kunnen zijn om een programma in staat te stellen de printer voor meer dan 1 formulier aan zich te binden: wie vier formulieren achter elkaar , dwz. ongestoord door overige programma's geprint wil hebben, mag de printer pas aan zich binden, als die vier formulierbeelden geheel geprepareerd zijn. Hiermee maakt men kunstmatig, dat de aanbiedende programma's in 'bursts' van meer formulieren aanbieden en het ligt dan ook voor de hand dat de buffer van de formulierprinter groter gekozen wordt, viz. groot genoeg om in enen op te kunnen vangen, wat een programma in enen aan zou willen bieden. We zijn echter niet van plan om een programma meer dan 1 formulier tegelijkertijd aan te laten bieden, en dus krijgen de individuele programma's een prive-buffer van 1 formulier.

Aldus besloten rijst de vraag, hoe groot we de buffer van de formulierprinter zullen kiezen. Minstens 1 formulier, nl. het formulier, dat gedrukt wordt. Maar 1 formulier is een beetje weinig: dit betekent immers, dat na voltooiing van dit formulier de buffer gegarandeerd leeg is en dat voor de coordinator een morele haaststituatie zou ontstaan als je de printer tussen de formulieren niet wilt laten stokken. Die morele haastsituatie kan aanmerkelijk verlicht worden door de formulierprinter met een toonbank van twee formulieren te laten werken: die worden dan wiggel-waggel bedreven.

Met de beslissing, dat de formulierprinter over meer dan een 1-formuliersbuffer zal beschikken, is een wezenlijke uitbreiding gedaan: dit betekent, dat aanbod op de toonbank onder controle van een vulwijzer zal moeten geschieden.(waar we anders niet over hoefden te praten) en dat meer dan 1 programma tegelijkertijd een formulier zou kunnen willen aanbieden en dat we zorgen moeten dat er dan geen ongewenste interferentie op treedt. Breiden we nu de omvang van de formulierprinterbuffer nog uit, dan verandert er aan de structuur van de programma's niets meer, alleen geheugenindeling en de waarde van een naar constantes. De vraag is, of we de formulierprinter een grotere buffer willen geven. Ik geloof van niet.

De overweging is de volgende: als de formulierprinter een grote eigen buffer heeft, die ongeveer leeg is en als nu alle programma's ongeveer tegelijkertijd een formulier aanbieden, dan wordt deze buffer grotendeels gevuld en ligt daarmee voor lang vast, welke programma's door de printer in successie geholpen zullen worden. Dit volgens de regel "wie lang geleden het eerste kwam, zal later het eerste malen". Heeft de formulierprinter maar een kleine buffer, dan is deze ongeveer onmiddellijk vol en de overige programma's staan op één seinpaal te wachten. Zodra er dan weer ruimte in de buffer van de formulierprinter komt, dan mag de coordinator beslissen wie een pagina mag aanbieden. Door deze grotere vrijheid van de coordinator zal het bij kleinere buffer makkelijker zijn om prioriteit te geven aan bepaalde programma's.

We hebben nog een ander ding overwogen, maar we menen tot de conclusie gekomen te zijn, dat je dat beslist niet doen moet; de coordinator heeft dan namelijk nog minder te vertellen. Je kunt als volgt redeneren: ieder programma heeft zelf (van de programmeur) enig idee van de omvang van zijn informatieexplosies, die naar de printer moeten. Laat het daarom een locale buffer introduceren ter grote van een dergelijke "burst' -afgerond op hele formulieren uit de aard der zaak. Nu kan je het aanbieden van een pagina ook anders spelen: inplaats van transport van de informatie -of van een descriptor, wat in dit verband op hetzelfde neer komt- kan je ook de in een ALGOL programma gevormde formulierinhoud via een kettingrijgerij aanhaken aan de ketting van het"nog te printen spul". Dit heeft de organisatorische complicatie, dat na voltooiing van deze printhandeling dit heugelijk feit niet neutraal betekent "de printer is weer vrij", maar dat dit doorgemeld moet worden naar het programma, in welks locale buffer hierdoor weer een formulierplaats vrijgekomen is. Veel erger is echter, dat wanneer één aantal programma's samen in de machine zit, die met zijn allen outputbound zijn door de printer, zij geholpen worden naar evenredigheid van hun eigen bufferomvang: het aanvragen van een grote locale buffer is dan een manier om je latere mogelijke compagnons in de machine een hak te zetten. Met deze conclusie zijn we erg blij, want om organisatorische redenen hadden we er nl. al niet zo'n zin in.

Het volgende, dat besloten moet worden is in welke vorm de printer tussen de formulieren achtergelaten wordt. De X8 heeft geen mogelijkheid om te testen, wanneer de perforatie voorbijkomt. Ik neem aan, dat tussen formulieren de printer achter gelaten wordt met het papier in de positie gereed voor de bedrukking van de eerste regel. De formulierprinter mag van deze toestand uitgaan maar moet na gebruik van de printer dus ook zorgvuldig deze toestand achterlaten! En dit geldt nu niet alleen van de formulierprinter, maar het geldt voor elk programma, dat "tussendoor" van de snelle regeldrukker gebruik wil maken.

Op dezelfde manier moeten we beslissen, wat we zullen doen met de IFT (= Inflop) en AFT (= Acflop) van de printer. Wij hebben ons hierbij door de controlemogelijkheid laten leiden. (Terwijl ik dit verslag tik, merk ik, dat we ook anders te werk hadden kunnen gaan; ik beschrijf nu eerst, wat we gedaan hebben). We zijn er van uitgegaan, dat de formulierprinter gebruik zou maken van de terugmeldingen, die met het ophogen van IFT gemeld worden; we hebben gesteld, dat het printen van een formulier, voor zover het de formulierprinter aangaat, beschouwd moet worden als een onscheidbaar geheel, dat in zijn geheel fout of in zijn geheel goed gaat. Mogelijke reactie op een fout moet dus kunnen zijn: het hele formulier opnieuw proberen te printen. Verder hebben we aangenomen, dat er tussen opeenvolgende formulieren geen vorm van "overlap" zou zijn: de formulierprinter begint pas aan de regels van het volgende formulier, als het vorige formulier geheel geprint en gecontroleerd is. Dit impliceert dus, dat tussen de formulieren het startmagazijn van de snelle regeldrukker even helemaal leeg is.

Op grond hiervan hebben we gesteld, dat de nuchtere toestand van de regeldrukker zijn zou AFT = IFT = 0; AFT is duidelijk, er hangen geen startopdrachten meer; het idee van IFT = 0 kan uitgelegd worden als "er is niets meer te melden".

Nadat dit besloten is, moet worden afgesproken, in wat voor staat de formulierprinter de bijbehorende geheugenplaatsen, leegwijzer en startmagazijn zal aantreffen. Het leek ons verstandig om daaromtrent geen enkele veronderstelling te maken. Dit betekent, dat bij de aanvang van een formulier de formulierprinter begint met een leegwijzer in te vullen (op iets, wat er waarschijnlijk nog staat van het vorige formulier, maar dat doet er niet toe). We nemen zelfs aan, dat de gegevens in het startmagazijn er niet meer zullen staan, hoewel die, zie onder, bedoeld zijn om tijdens het formulierprinten niet te veranderen.

Tenslotte moeten we beslissen, waar het regelbeeld zal staan, dat door de regeldrukker geprint zal worden. Dit kost ons 32 woorden per regel en we moeten totaal 4 regels kunnen opbergen. We hebben besloten hier 128 woorden vast voor te reserveren en dus op het allerlaatst de informatie wel lijfelijk te transporteren. De overwegingen, die hier toe geleid hebben, waren de volgende. Als een pagina met regels nog op de kernen staat, staat hij "ergens", dwz. op een vrij willekeurige kernpagina; als de pagina met regels inmiddels op de trommel gedumpt is en we hem voor gebts halen met het standaard mechanisme -en dat willen we natuurlijk- dan komt hij ergens. Maar informatie, die door de printer asynchroon uit het geheugen geslurpt wordt, moet staan op een uiters heilige plaats. Zouden wij de pagina, zoals hij in het kerngeheugen staat of komt, heiligen, dan zou dat impliceren, dat we eventueel midden in het kerngeheugen voor vrij lange tijd een geheiligde pagina zouden hebben. De beperkingen, die daaruit voortvloeien, trokken ons niet aan. Vandaar deze beslissing.

Nu geven we de voorlopige versie van de programma's; zij zijn geschreven met inbegrip van de controle, de detectie, als het goed is gegaan. Het leek ons nu wat vroeg om te gaan uitmelken, wat we moeten doen, als deze controle ALARM geeft.

Relevant grootheden zijn:

integer FPN; aantal formulieren in FP-buffer (Formulierprinter buffer); dit is een constante, waarschijnlijk dus gelijk 2
integer FPLW, FPVW; dit zijn leegwijzer, resp. vulwijzer van de FP-buffer deze moeten door het tandenpoetsen aan elkaar gelijk gemaakt worden (door het tandenpoetsen van de formulierprinter); zij worden modulo FPN in het bedrijf opgehoogd.

Om der wille van de beschrijving voer ik in een nieuw type array, een zg TPV array elementen waarvan fungeren als identifier van een array van 512 integers; ik zal deze elementen met dubbele indicering aanduiden.

Globaal voor formulierprinter en elk ALGOL programma hebben we de toonhank, alias de formulierprinterbuffer
TPV array FPB[0:4*FPN-1]; deze factor 4 komt omdat we per formulier vier pagina's nodig hebben. Een ALGOL-programma bouwt zijn formulier op in het locaal formulier
TPV array LF[0:3];

Voor de reductie modulo iets gebruiken we de integer procedure REM.

Synchronisatie tussen een ALGOL-programma en de formulierprinter wordt gespeeld via twee globale seinpalen:
integer FPL,FPV; deze betekenen het aantal lege, resp. volle formulieren in FPB. Aanvankelijk (dwz, bij tandenpoetsen van de formulierprinter wordt FPL:= FPN; FPV:= 0 uitgevoerd).

Aanbod van een formulier aan de formulierprinter geschiedt door het volgende stukje programma (hier hebben we nog een willekeurige locale integer, zeg i nodig):

```
begin X8 doof; P(FPL);
      for i:= 0,1,2,3 do begin FPB[4*FPVW + i]:= LF[i]; LF[i]:= maagdelijk end;
      FPVW:= REM(FPVW + 1, FPN);
      V(FPV); X8 horend
end
```

Commentaar: de X8 wordt hier doof gemaakt om te zorgen dat dit aanbod onder controle van een unieke vulwijzer plaats vindt; noninterferentie van de ophoging van FPVW is hiermee ook verzekerd. Dit had "zuiniger" onder controle van een speciale uitslutingsseinpaal gekund, zuiniger in de zin, dat we dan minder tijdsrestricties opleggen. Dit accevietje duurt echter gegarabdeerd zo kort, dat even doof minder werk is. Het feit, dat ook de P- en V-operatie in diifheid uitgevierd worden, is logisch niet essentieel.

Met "maagdelijk" is bedoeld de constante waarde van een TPV, waarvoor nog geen pagina gereserveerd is. In dit geval wordt het dus een TPV, waarvoor geen pagina meer gereserveerd is.

De formulierprinter als hieronder geprogrammeerd neemt van de printer alleen maar aan, dat IFT en AFT beide = 0 zijn. Het startmagazijn en de leegwijzer van de printer moeten dus passend geprepareerd worden. Ik gebruik hier de integer procedure address ter beschrijving van "het adres van".

Naast de genoemde globale grootheden heeft de formulierprinter toevang tot de integer "leegwijzer", de grootheid, [illegible] onder controle waarvan de printer asynchroon de startopdracht aanhaalt. Verder is er een globale seinpaal "PV", te weten printer vrij; deze seinpaal stelt ons in staat desgewenst de printer tussendoor ook even op een andere manier te gebruiken.

De tekst van de formulierprinter wordt ongeveer als volgt:

```
formulierprinter:
begin integer leegwijzer, i, j, vulwijzer;
      integer array startmagazijn [0:7], regelmagazijn [0:127];
      [illegible]
      P(FPV, PV);
      leegwijzer:= address(startmagazijn[7]);
      for i:= 0,1,2,3 do startmagazijn [2*i]:= address(regelmagazijn[32*i]) + cons;
      i:= vulwijzer:= 0;
nieuwe regel:
      if  3 < i then  begin P(IFT printer);
      [illegible] if startmagazijn[2*vulwijzer + 1] ≠ correct then  goto ALARM  end;
      if i ≤ 63 then
      begin  for j:= 0 step 1 until [illegible] 31 do
                regelmagazijn[32* vulwijzer + j]:=
                          FPB[4*FPLW + entier((i+0.5)/16)][32*REM(i,16) + j];
             V(AFT printer);
             vulwijzer:= REM(vulwijzer + 1, 4)
      end;
      if i < 68 then goto nieuwe regel;
      for i:= 0,1,2,3 do vrijgave pagina(FPB[4*[illegible] FPLW + i]);
      FPLW:= REM(FPLW + 1, FPN);
      [illegible] V(FPL, PV);  goto formulierprinter
end
```

Opm. het begint met initialiseringen, als het startmagazijn en het regelmagazijn vast gekozen worden, dan zijn dit altijd dezelfde constanten, die hier invevuld moeten worden. Er is geen bezwaar tegen, dat dit van formulier tot formulier zou weizigen. De test "3 < i is om de eerste vier keer de controle te onderdrukken, de test "[illegible] i ≤ 63" om alleen maar feitelijk aan te bieden, als er nog iets aan te bieden valt.
De procedure "vrijgave pagina" krijgt een tpv als argument mee; het resultaat is, dat de trommelpagina, die via deze TPV toegankelijk was, nu definitief aan de voorraad vrijen wordt toegevoegd. Zou je zoiets als dit vergeten, dat zou de trommel dichtgroeien met oude paginabeelden.

De conventie dat de formulierprinter de printer met IFT = 0 aantreft, wat we in verband met de controle gedaan hebben, lijkt me bij nader inzien questieus. Nu dwing je min of meer elk ander gebruik nodeloos in dit keurslijf. Beter lijkt het me om als neutrale toestand AFT = 0 en IFT = 4 (de omvang van het startmagazijn) te kiezen; dit is niet in tegenspraak, dat de formulierprinter pas na volledige controle overgaat op het volgende formulier. De formulierprinter kan aan het einde, na de controle 4 maal V(IFT printer) doen! Wel ongebruikelijk, maar moeten we er niet aaan wennen?