Programmeren in C

m.b.v. programma-structuur-diagrammen.

Programmastructuurdiagram (PSD)

Een aantal jaren geleden is er een verhitte discussie geweest onder programmeurs over het gestructureerd programmeren van software.
Voordat we aangeven waar de discussie over ging, wordt eerst het begrip ***gestructureerd programma*** verduidelijkt. Een gestructureerd programma is een programma dat we gemakkelijk kunnen begrijpen. Dit kunnen we bereiken door het programma op te bouwen uit drie programmacomponenten:

– opeenvolging;
– selectie (keuze);
– herhaling.

Als we deze componenten gebruiken, wil dat nog niet zeggen dat het programma gestructureerd is. De programmacomponenten moeten we wel op een gedisplineerde wijze gebruiken.
De genoemde discussie ging over de vraag of er sprongopdrachten mogen worden toegepast in een gestructureerd programma. De heren Bohm en Jacopini hebben met een stelling bewezen dat elk programma dat sprongopdrachten gebruikt, omgezet kan worden in een programma dat alleen de programmacomponenten
opeenvolging, ***selectie*** en ***herhaling*** gebruikt. Het is opmerkelijk dat deze onderzoekers niet hebben verteld hoe deze omzetting moet gebeuren.
Het resultaat van de discussie is het gebruik van *programmastructuurdiagrammen*.
In de inleiding van deze standaard wordt gezegd dat het programma gestructureerd is als we de programmacomponenten en combinatieregels gebruiken.

De programmacomponenten zijn:
– serieel (sequentie);
– parallel;
– herhaling;
– selectie;
– uitbreek.

Functionele decompositie:

Een ander belangrijk onderdeel van gestructureerd programmeren is het toepassen van een *top-downbenadering*. De top-downbenadering houdt in dat we het probleem waarvoor we een programma moeten schrijven, in steeds kleinere deelproblemen splitsen, totdat we het deelprogramma rechtstreeks kunnen omzetten in een computerprogramma. Dit noemen we ook wel ***functionele decompositie***.

Aspecten van het PSD-diagram

Het PSD-diagram is vastgelegd in de NEN 1422. In deze norm komen de volgende zaken aan de orde over het PSD-diagram:
– de omschrijving van programmacomponenten;
– de manier waarop programmacomponenten kunnen worden gecombineerd;
– de specificatie van de programmacomponenten;
– de grafische voorstelling van de programmacomponenten.

1.4.1 *Algemene opbouw en grafische voorstelling*

Zoals in de inleiding al is gezegd, is een gestructureerd programma samengesteld uit verschillende programmacomponenten.
Nu volgt een algemene omschrijving van de opbouw van een programmacomponent.

Programmacomponent

Een programmacomponent is dus een bouwsteen van het programma als geheel. Het bestaat uit:

– één besturingsdeel;
– één of meer opdrachtdelen.

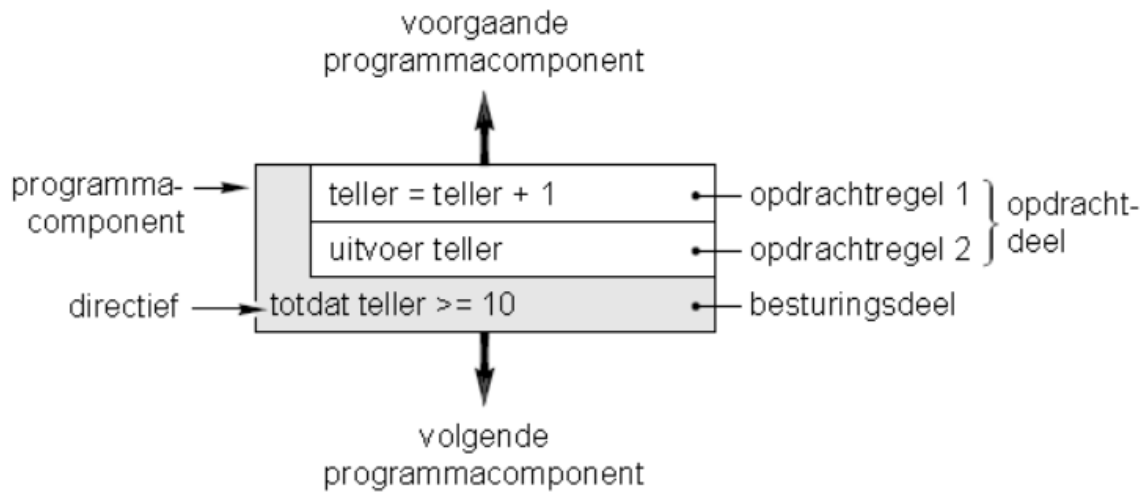
De opdrachtdelen van een programmacomponent worden geactiveerd of gedeactiveerd door het besturingsdeel. Het besturingsdeel bepaalt dus de uitvoeringsvolgorde van de opdrachtdelen. Hiervoor wordt een directief gebruikt in combinatie met één of meer voorwaarden. Een *directief* is een pseudo-opdracht die een aanwijzing geeft voor de vertaling van het programmastructuurdiagram in een programma.

De pseudo-opdracht wordt niet omgezet in een opdracht van een programmeertaal en is dus eigenlijk geen echte opdracht maar een *pseudoopdracht*.

Niet alle besturingsdelen maken gebruik van een directief.
Voorbeelden van directieven zijn:

– zolang;
– totdat;
– als.

In figuur 1.33 zien we een programmacomponent, hier een herhaling, weergegeven met een programmastructuurdiagram. Het besturingsdeel is geaccentueerd met een grijze achtergrond. Het besturingsdeel activeert de twee opdrachtregels net zo lang totdat het resultaat van de voorwaarde *waar* is. In dit geval betekent dat de variabele *teller* groter of gelijk moet zijn aan tien.
De programmacomponent wordt geactiveerd nadat de voorgaande programmacomponent is afgewerkt. Op het moment dat de voorwaarde *waar* wordt, wordt de programmacomponent afgesloten en wordt er verdergegaan met de volgende programmacomponent.



Figuur 1.33 Herhalingscomponent

Opdrachtcomponent

Een van de basiscomponenten is de ***opdrachtcomponent*** en die bestaat alleen uit een opdrachtdeel. Het besturingsdeel is weggelaten, omdat de opdrachtcomponent altijd eenmaal wordt uitgevoerd, onafhankelijk van allerlei voorwaarden. De opdrachtcomponent is het onderdeel waarin de operaties worden genoteerd om tot een bepaald resultaat te komen.

De opdrachtcomponent mag ook de volgende inhouden bevatten:
– leeg zijn;
– commentaar bevatten;
– een verwijzing bevatten naar een andere plaats in het diagram of naar een ander diagram.

De opdrachtcomponent wordt voorgesteld door een rechthoek.
In de rechthoek wordt met tekst de opdracht weergegeven die op dat moment moet worden uitgevoerd.

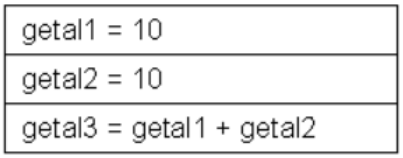
In figuur 1.35 zien we een opdrachtcomponent waarbij een getal wordt toegekend aan een variabele. Dit wordt in dit geval dus één keer gedaan.

Figuur 1.35 Voorbeeld van opdrachtcomponent

Sequentiecomponent

Een algoritme bestaat vaak uit verscheidene handelingen die na elkaar moeten worden uitgevoerd. Om dit te realiseren, wordt er gebruikgemaakt van een *sequentiecomponent*. De sequentiecomponent is opgebouwd uit twee of meer onder elkaar geplaatste opdrachtcomponenten. Zie figuur 1.36. Dit is dus een rij van opdrachtcomponenten. Deze worden dan na elkaar uitgevoerd. De opdrachtcomponenten afzonderlijk worden ook nu natuurlijk maar één keer uitgevoerd.
Het voorbeeld in figuur 1.37 bevat een PSD bestaande uit een sequentiecomponent. Daarbij worden aan twee variabelen waarden toegekend en die waarden worden daarna opgeteld.

Figuur 1.36 Sequentiecomponent



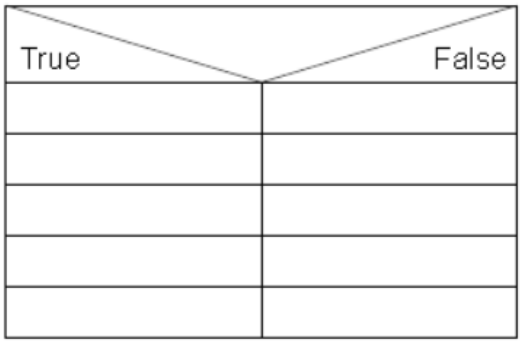
Figuur 1.37 Voorbeeld van sequentiecomponent

Parallelle component

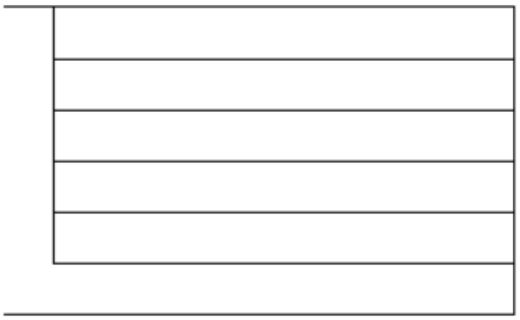
In een sequentiecomponent kan er maar één opdrachtcomponent gelijktijdig actief zijn. Bij een *parallelle component* kunnen er tegelijkertijd verscheidene opdrachtcomponenten actief zijn. De parallelle component is opgebouwd uit twee of meer naast elkaar geplaatste opdrachtcomponenten. Deze worden tegelijkertijd uitgevoerd. Als de laatste opdrachtcomponent is uitgevoerd, stopt

Combineren van componenten

In de volgende onderdelen worden de overige programmacomponenten besproken. Deze componenten kunnen we combineren tot een programma. Het combineren is wel gebonden aan één regel: het opdrachtdeel van een component moet worden gecombineerd met een besturingsdeel. In figuur 1.39 en figuur 1.40 zien we twee voorbeelden van deze regel.



Figuur 1.39 Combinatie van sequentie en keuze



Figuur 1.40 Combinatie van sequentie en herhaling

1.4.2 Selectiecomponenten

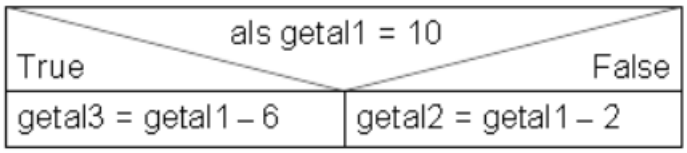
De selectie-programmacomponenten gebruiken we voor het kiezen of een bepaalde groep van opdrachten uitgevoerd moet worden of niet, afhankelijk van het resultaat van een voorwaarde. Er zijn drie verschillende selectiecomponenten, namelijk voor:

– tweevoudige selectie;
– enkelvoudige selectie;
– meervoudige selectie.

Tweevoudige selectie

De *tweevoudige-selectiecomponent* bestaat uit twee opdrachtdelen en een besturingsdeel. Het besturingsdeel activeert of deactiveert één van de opdrachtdelen, afhankelijk van het resultaat van de voorwaarde.

Figuur 1.41 Symbool van tweevoudige selectie

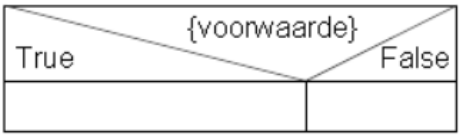


Figuur 1.42 Voorbeeld van tweevoudige selectie

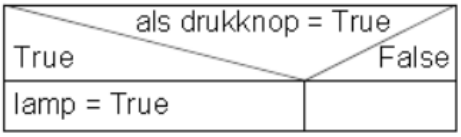
De grafische voorstelling van de tweevoudige-selectiecomponent zien we in figuur 1.41 en figuur 1.42. Het besturingsdeel is opgedeeld in drie vlakken. In het bovenste vlak noteren we de voorwaarde die gebruikt wordt voor het maken van de keuze.
Het middelpunt van de drie vlakken symboliseert een splitsing van het opdrachtdeel in twee velden: de keuzemogelijkheden.
Het eerste veld wordt uitgevoerd, als het resultaat van de voorwaarde overeenkomt met de waarde die genoteerd staat in het vlak dat aangrenzend ligt in het besturingsdeel. De opdrachten in het andere veld worden uitgevoerd, als het resultaat van de voorwaarde overeenkomt met de waarde in het daar aangrenzende vlak.
De velden kunnen we zien als afzonderlijke opdrachtdelen. Het middelpunt hoeft niet altijd in het midden van de programmacomponent te liggen.

Enkelvoudige selectie

De *enkelvoudige-selectiecomponent* bestaat uit één opdrachtdeel en een besturingsdeel. Het besturingsdeel activeert of deactiveert het opdrachtdeel, afhankelijk van het resultaat van de voorwaarde. De component bestaat eigenlijk uit twee opdrachtdelen. Eén van de opdrachtdelen is leeg.
De enkelvoudige selectie is eigenlijk een bijzondere vorm van de tweevoudige selectie. De grafische voorstelling komt dan ook geheel overeen met de tweevoudige selectie. Zie figuur 1.43 en figuur 1.44. Het tweede opdrachtdeel wordt echter niet gebruikt.



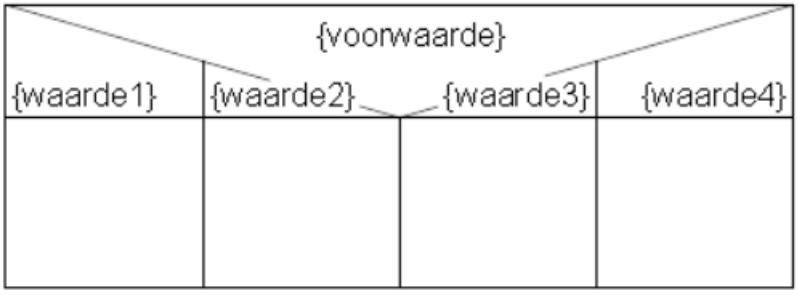
Figuur 1.43 Symbool van enkelvoudige selectie



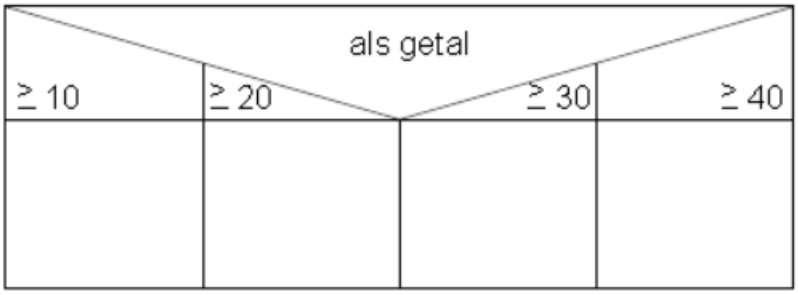
Figuur 1.44 Voorbeeld van enkelvoudige selectie

Meervoudige selectie

De *meervoudige-selectiecomponent* bestaat uit twee of meer opdrachtdelen en een besturingsdeel. Het besturingsdeel activeert of deactiveert één of meer opdrachtdelen, afhankelijk van het resultaat van de voorwaarde.
De grafische voorstelling van de meervoudige selectie is op één punt afwijkend van de tweevoudige selectie. Omdat we de meervoudige selectie gebruiken voor het maken van een keuze uit verscheidene mogelijkheden, wordt het opdrachtdeel gesplitst in verscheidene velden. Zie figuur 1.45 en figuur 1.46. Het aantal velden is afhankelijk van de keuzemogelijkheden. Welke opdrachtdelen worden uitgevoerd, is afhankelijk van het resultaat in het besturingsdeel.



Figuur 1.45 Symbool van meervoudige selectie



Figuur 1.46 Voorbeeld van meervoudige selectie

1.4.3 Herhalingscomponenten

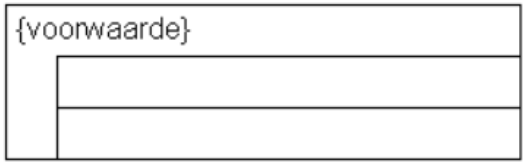
Vaak moeten bepaalde programmaonderdelen een aantal keren worden herhaald. Dit gebeurt door middel van de herhalingscomponent. Er zijn drie verschillende herhalingscomponenten, namelijk voor:

– herhaling met vergelijking vooraf;
– herhaling met vergelijking achteraf;
– continue herhaling.

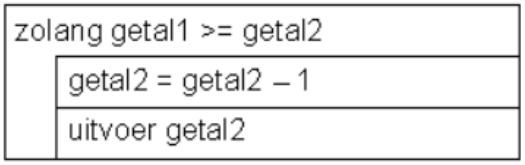
Het verschil tussen deze herhalingscomponenten wordt weerspiegeld in hun grafische voorstellingen.

Herhaling met vergelijking vooraf

Een *herhalingscomponent met vergelijking vooraf* bestaat uit een opdrachtdeel en een besturingsdeel. Het besturingsdeel bestaat uit een voorwaarde. Het resultaat daarvan geeft aan of het opdrachtdeel nul, één of meer keren wordt uitgevoerd.
In de grafische voorstelling kunnen we het besturingsdeel in gedachten opdelen in twee rechthoeken, waarvan er één verticaal is getekend en de andere horizontaal. Zie figuur 1.47 en figuur 1.48. De horizontale rechthoek is de plaats waar we de voorwaarde van het herhalen kunnen noteren. Hij is geplaatst voor de opdrachtdelen die moeten worden herhaald. De verticale rechthoek loopt langs de opdrachtdelen die worden herhaald, afhankelijk van de voor afgestelde voorwaarde.



Figuur 1.47 Symbool van herhaling met vergelijking vooraf

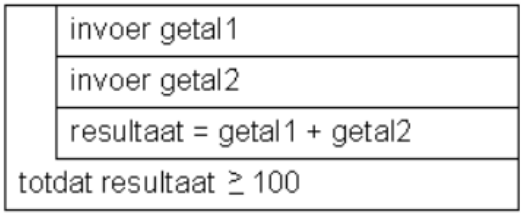


Figuur 1.48 Voorbeeld van herhaling met vergelijking vooraf

Herhaling met vergelijking achteraf

Een *herhalingscomponent met vergelijking achteraf* bestaat uit een opdrachtdeel en een besturingsdeel. Het besturingsdeel bestaat uit één voorwaarde. Het resultaat daarvan geeft aan of het opdrachtdeel één of meer keren wordt uitgevoerd.
De grafische voorstelling is bijna gelijk aan die van de andere herhaling. Alleen is de voorwaarde nu achteraf geplaatst. Zie figuur 1.49 en figuur 1.50.

Figuur 1.49 Symbool van herhaling met vergelijking achteraf



Figuur 1.50 Voorbeeld van herhaling met vergelijking achteraf

1.4.4 Uitbreken

Een programma is meestal opgebouwd uit programmaonderdelen. Een programmaonderdeel wordt uitgevoerd als het voorliggende onderdeel klaar is. Dit wordt bepaald door de voorwaarde in de besturingscomponent. Een programmaonderdeel moet dus geheel worden uitgevoerd, anders kan er geen ander programmaonderdeel worden uitgevoerd.
Er zijn situaties dat bepaalde programmaonderdelen vroegtijdig moeten worden beëindigd. Dit wordt dan gerealiseerd met een *uitbreek-programmacomponent*. Zie figuur 1.52 en 1.53. Het uitbreken wil zeggen dat genoemde programmaonderdelen
en de programmacomponenten die zich daarin bevinden,
onmiddellijk worden beëindigd. In het symbool moet dus duidelijk worden aangegeven welk programmaonderdeel moet worden beëindigd.

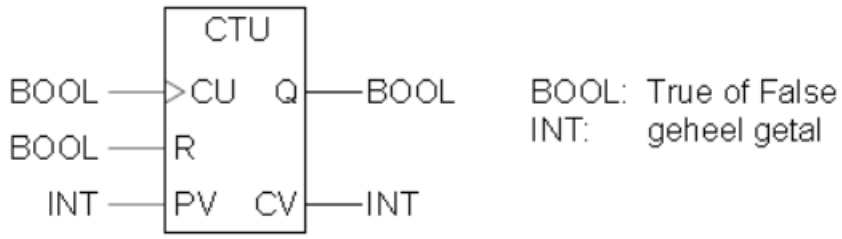
Figuur 1.52 Symbool van uitbreken

Figuur 1.53 Voorbeeld van uitbreken

1.4.5 Voorbeelden van PSD's

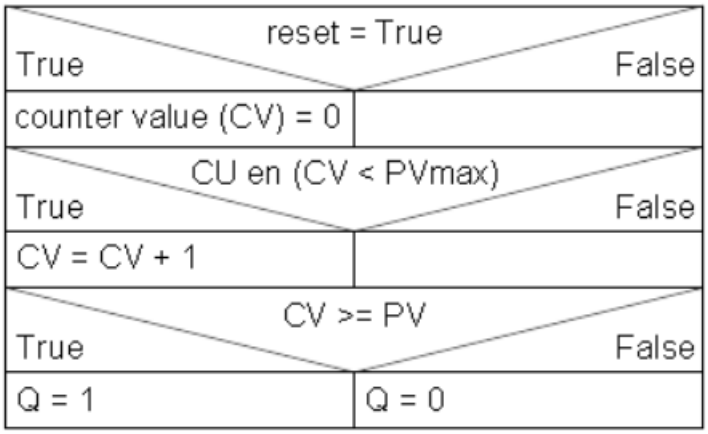
Voorbeeld: up-counter

In dit voorbeeld wordt de werking van de up-counter uitgelegd aan de hand van het PSD van deze counter. Het symbool van een up-counter zien we in figuur 1.54.



Figuur 1.54 Symbool van up-counter

Een counter wordt gebruikt voor bijvoorbeeld het tellen van producten of van toestanden die voorkomen in een programma. Een counter is daarom uitgevoerd met een ingang waarmee een variabele kan worden verbonden die geteld moet worden, de *CU-ingang* (counter up).
Op het moment dat aan deze ingang een opgaande flank verschijnt, wordt de counterwaarde met één verhoogd. De counterwaarde wordt in het symbool aangegeven met *CV* (*counter value*). Deze waarde kunnen we met de resetingang
(*R*) naar nul zetten. De *PV-ingang* is de *preset value* van de counter. Op het moment dat de *PV* gelijk is aan de *CV*, wordt de uitgang Q *waar*.
In figuur 1.55 zien we de PSD van de counter. PVmax is de maximale teller waarde die de PLC kan bevatten.



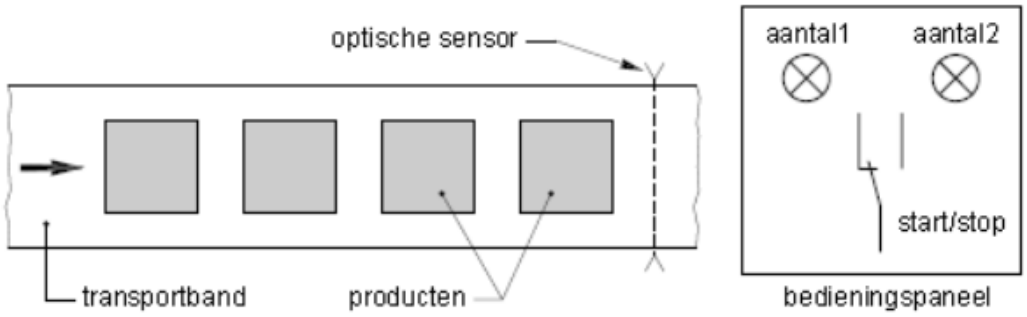
Figuur 1.55 PSD van up-counter

Voorbeeld: transportband

In dit voorbeeld worden producten geteld die op een transportband worden getransporteerd. Er wordt in het programma gebruikgemaakt van een counter. De transportband moet stoppen op het moment dat er twintig producten zijn geteld. In de tussentijd moet het aantal zichtbaar worden gemaakt met twee signaallampjes op een bedieningspaneel. De signaallampjes hebben de naam *aantal1* en *aantal2*. Zie figuur 1.56.

Er wordt gebruikgemaakt van een digitale code. Zie tabel 1.3.

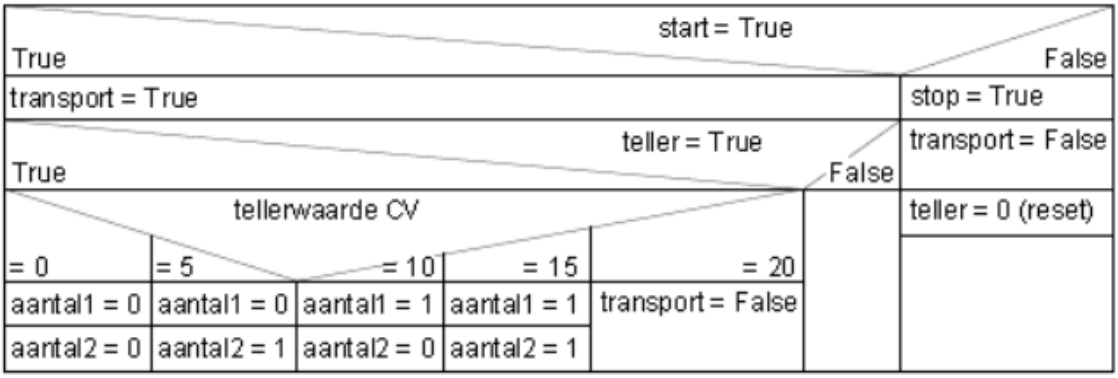
Ook tussentijds kan de transportband worden gestopt. De transportband stopt en de teller moet dan worden gereset.
Figuur 1.57 geeft de PSD.



Figuur 1.56 Opstelling transportband

TABEL 1.3 OVERZICHT CODE

Waarde	Aantal1	Aantal2
0	0	0
5	0	1
10	1	0
15	1	1

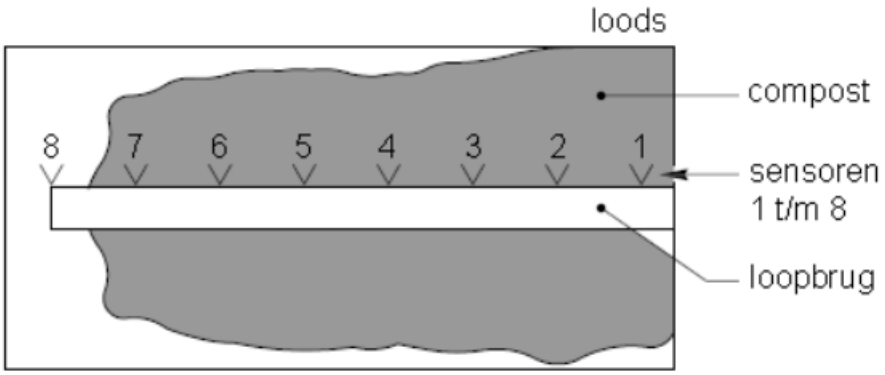


Figuur 1.57 PSD van transportband

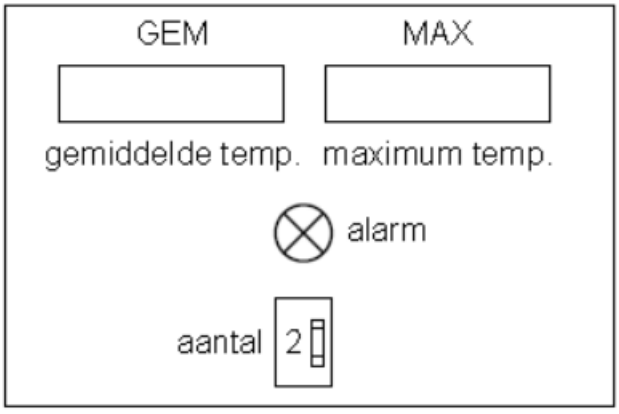
Voorbeeld: compostopslag

Bij compost kan door druk en vochtigheid warmte ontstaan, waardoor het materiaal spontaan gaat ontleden. Deze ontleding zorgt voor extra warmte en kan een kettingreactie op gang brengen. Door de kettingreactie kan een grote warmteontwikkeling ontstaan, waardoor de compost spontaan kan gaan broei ontbranden. Dit verschijnsel noemen we *broei*.

Broei kunnen we tegengaan door de compost droog te houden en de druk niet te hoog laten oplopen. Ook door de compost uit elkaar te halen kunnen we broei verminderen.
Een belangrijk gegeven is de temperatuur van de compost. De compost wordt opgeslagen in een loods met een loopbrug boven de compost. Aan deze loopbrug zijn sensoren bevestigd, die we handmatig in de compost kunnen prikken om de temperatuur in de compost te meten. De loods is niet altijd gevuld. Er is dus een variabel aantal sensoren die de temperatuur meten. In figuur 1.58 zien we een overzicht van de opslagloods.

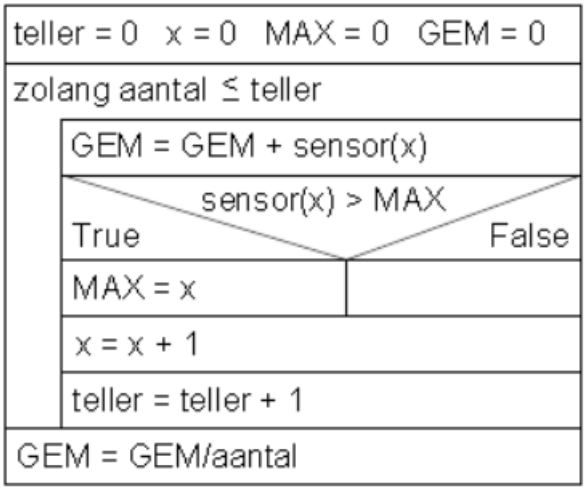


Figuur 1.58 Bovenaanzicht van loods voor opslag van compost



Figuur 1.59 Bedieningspaneel

Het bedieningspaneel zien we in figuur 1.59. Het bevat een duimwielschakelaar waarmee het aantal sensoren die moeten meten, wordt aangegeven. Dat is het aantal sensoren vanaf sensor 1. Ook wordt de gemiddelde temperatuur en de maximale temperatuur weergegeven met 7-segmentsdisplays. Er gaat een alarm af als er een maximale temperatuur is bereikt. Het alarm bestaat uit een claxon en een lamp op het bedieningspaneel.
In figuur 1.60 zien we de PSD van een programma. Dit programma kan de besturing realiseren die een alarm geeft als de maximumtemperatuur wordt bereikt en die aangeeft door welke sensoren deze maximumtemperatuur wordt vastgesteld. Hierdoor kan de procesoperator ervoor zorgen dat de compost uit elkaar wordt gehaald om broei tegen te gaan.



Figuur 1.60 PSD van compost-alarm

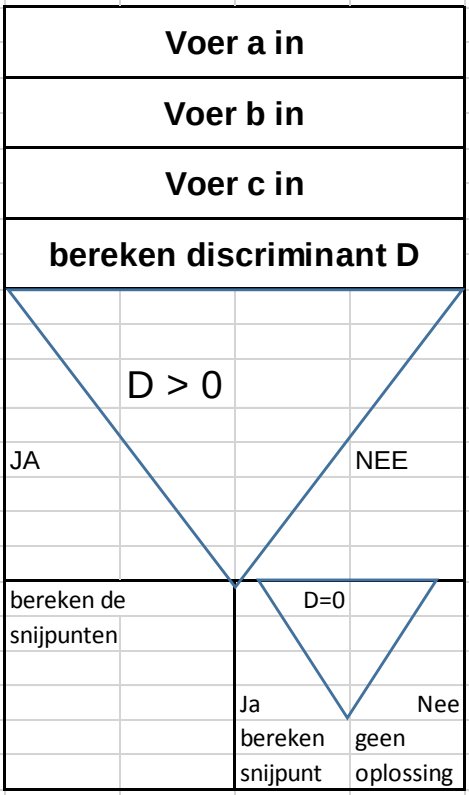
De variabele *aantal* in het diagram is het aantal ingestelde sensoren. De variabele
teller is een hulpvariabele die vaak wordt gebruikt bij een herhaling. Deze variabele houdt het aantal keren bij dat een herhaling is uitgevoerd.
Een aparte variabele is *sensor(x)*. Hiermee wordt aangegeven dat het een sensor is uit de groep van acht sensoren.

Opdrachten:

Programmeren met Excel:

abc formule met Excel.

a	1,00
b	5,00
c	6,00
D	1,00
snijpunt x1	-2
snijpunt x2	-3
snijpujnt 1 =2	ONWAAR



Schrijf programma in de taal C.

Nu een uitbreiding:

Er geldt dat a<>0 (waarom ?) Bouw deze uitbreiding in je programma.

Pneumatiek Schakelalgebra

Waarheidstabel

a	b	a of b	a en b	a of b	a en b
0	0	ONWAAR	ONWAAR	0	0
0	1	WAAR	ONWAAR	1	0
1	0	WAAR	ONWAAR	1	0
1	1	WAAR	WAAR	1	1

=OF(cel;cel) geeft als antwoord waar of onwaar

=EN(cel;cel) geeft als antwoord waar of onwaar

=ALS(cel;1;0) geeft als antwoord 1 of 0

=ALS(en(cel;cel);1;0) geeft als antwoord 1 of 1

opdracht 1	Maak van de volgende formules een: Waarheidstabel, een logisch schema en een pneumatisch schema.
1,1	$a \cdot \overline{b} = u$
1,2	$a.b.c = u$
1,3	$a + b.c = u$

≥ of-functie

& en-functie

Maak de opdrachten 1.1 t/m 1.3 in Excel.

Voorbeelden programma's in C

```c
/*
 *  Hoofdstuk 1 - programma 1 : 1-1.c
 *  Druk 2 ingegeven getallen in omgekeerde volgorde af.
 */

#include <stdio.h>

int main(void)
{
int getal1, getal2;

    printf("Geef het eerste getal -> ");
    scanf("%e",&getal1);

    printf("Geef het tweede getal -> ");
    scanf("%d",&getal2);

    printf("\n");                                          /* een lege regel */

    printf("De twee getallen in omgekeerde volgorde zijn: %d   %d",
                                                              getal2, getal1);
   return 0;
}
```

```c
/*
 *      Hoofdstuk 2 - programma 3: 2-3.c
 *      Lees een rij positieve reeële getallen in (aantal onbekend)
 *    en bereken op het  einde het gemiddelde en druk het af.
 *      Gebruik een negatief getal als afsluitwaarde.
 */

#include <stdio.h>

int main(void)
{
float getal, som=0, gemiddelde;

        /* teller op 0 om straks het afsluitgetal niet mee te verwerken*/
        int teller=0;

        printf("Geef een reeks getallen in (sluit af met negatief getal) -> ");

        scanf("%f",&getal);

   while(getal > 0)
   {
            som+=getal;
            scanf("%f",&getal);
            teller++;
   }

        gemiddelde = som / teller;

        printf("Het gemiddelde van de %d getallen is = %.2f",teller,gemiddelde);

   return 0;
}
```