Programmeren – Blok A

http://www.win.tue.nl/~wstomv/edu/2ip05/

College 2

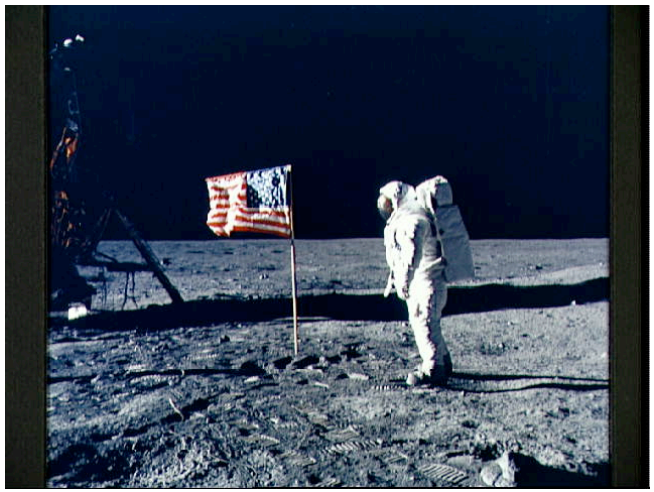
Tom Verhoeff

Technische Universiteit Eindhoven
Faculteit Wiskunde en Informatica
Software Engineering & Technology

Opmerkingen aan T.Verhoeff@TUE.NL

Wat was het doel?

Wat was het doel?



Stand van zaken

Groep	Stud.	GetalKlutser			Snoep			Mediaan		
		–	Rej.	OK	–	Rej.	OK	–	Rej.	OK
A B E	42	6	0	36	5	5	32	6	3	33
C D F	46	4	0	42	4	5	37	5	2	39
Totaal	88	10	0	78	9	10	69	11	5	69

Inzenden verplicht voor *huiswerkopgaven*!

Snoep 1: Goed of Fout?

```
13 begin { toestandsveranderingen }
14 ; readln ( K, C )
15 ;if (K > 0) and (C > 0)
16 then
17 begin
18 ; if C mod K = 0
19   then
20   begin
21 ;    Q := C div K
22 ; if (K > 0) and (C > 0) and (Q > 0)
23   then
24   begin
25 ;    writeln ( 'Yes')
26 ;    writeln ( Q )
```


Snoep 1: Goed of Fout?

```
27 ;    write ( 'Tik <return> om programma te verlaten: ' )
28   end
29   end
30
31   else
32   begin
33 ;    writeln ( 'No' )
34 ;    write ( 'Tik <return> om programma te verlaten: ' )
35   end
36 end
37 else
38 ; readln
39 end.
```


Snoep 2: Goed of Fout?

```
 3   { Berekend of het mogelijk is dat elk kind evenveel snoepjes krijgt
 4     en hoeveel snoepjes dat dan zijn, hierbij zijn het aantal kinderen
 5     en snoepjes variabel. }
16 ; if K > 0 then begin { check of berekening nodig is ( K > 0 ) }
17   ; if (C mod K = 0) then begin { snoepjes zijn eerlijk te verdelen }
18       Q := C div K
19     ; writeln ( 'Yes' )
20     ; writeln ( Q )
21     end else { snoepjes zijn niet eerlijk te verdelen }
22       writeln ( 'No' )
23   end else begin { K = 0 }
24     writeln ( 'Yes' )
25   ; writeln ( 0 )
26   end
```


Snoep 3: Goed of Fout?

```
12 Q := 0; {init Q}
13 readln (K, C);  {lees Kinderen en Snoep}
14 if (K <> 0) THEN begin; {Als er kinderen zijn:}
15   if (C mod K = 0) THEN {Als het eerlijk te verdelen is, zet het aantal snoep
16     Q := C div K;
17   if Q = 0 THEN {Als het niet eerlijk te verdelen is, output No}
18     writeln( 'No' );
19   if Q > 0 THEN begin; {Was het eerlijk te verdelen, output yes, en het aantal
20     writeln('Yes');
21     writeln(Q);
22     end;
23 end;
24 if (K = 0) THEN begin; {Als er geen kinderen waren}
25     writeln('No');
26 end;
```

Snoep 4: Goed of Fout?

```
15 ; if c >= 0 then      {Negatieve snoepjes gaat moeilijk}
16   q := c div k
17
18   ; if c mod k = 0 then    {Als er geen rest overblijft, Yes uitvoer}
19     begin
20       writeln ( 'Yes' )
21     ; writeln ( q ) end
22     else                    {Als er wel rest overblijft, No uitvoer}
23       writeln ( 'No' )
```


Snoep 5: Goed of Fout?

```
 8         var kinderen,snoep: integer;
 9 begin
10         readln(kinderen, snoep);
11         if (snoep > 0) and (kinderen > 0) then begin
12                 if  snoep mod kinderen = 0 then begin
13                         writeln('Yes');
14                         writeln(snoep div kinderen);
15                         end
16                 else writeln('No');      {snoep is niet deelbaar door kinderer
17         end
18         else if ((snoep > 0) and (kinderen = 0)) or
19                           ((snoep = 0) and (kinderen > 0)) then begin
20                 writeln('Yes');
21                 writeln(0);
22                 end
23         else
24                 writeln('No');
```


Snoep 6: Goed of Fout?

```
16 ;if k > 0 then
17   begin
18     q := c div k  { Deel aantal snoepjes door de aantal kinderen }
19
20     ;if c mod k = 0 then { Als er geen snoep overblijft na de deling dan is he
21       begin
22         writeln('Yes')
23         ;writeln( q )
24       end
25       else
26         writeln('No')
27   end
28   else
29     if k <= 0 then
30         writeln('No')
```


Snoep 7: Goed of Fout?

```
16   if ( i_kinderen <= 0 ) then i_kinderen := 1 ;
17
18   i_Q := i_snoep mod i_kinderen;
19   i_perkind := (i_snoep - i_Q) div (i_kinderen);
20
21   if ( ( i_Q = 0) and (i_perkind >0)) then begin
22     writeln ('Yes');   { Als het eerlijk verdeeld kan worden en ieder kind kr
23     writeln( i_perkind );  { Output snoepjes per kind }
24   end
25
26   else writeln ('No');
```

Snoep 8: Goed of Fout?

```
17 ; if ( k > 0 ) and ( c > 0 ) then { controleer of invoer groter is dan 0 }
18   begin
19     if c mod k = 0 then { controleer of er snoepjes overblijven }
20     begin
21       q := c div k           { wijs het aantal snoepjes per persoon toe }
22 ;     writeln( 'Yes' )       { schrijf 'yes' naar scherm }
23 ;     writeln( q )           { schrijf aantal snoepjes pp naar scherm }
24 ;   end else writeln( 'no' ) { schrijf 'no'naar scherm }
25 ; end
```


Snoep 9: Goed of Fout?

```
1  if K = 0 then begin { special case }
2  if C = 0 then { possible }
3  Q := 0
4  else { impossible }
5  Q := -1
6  end
7  else begin { K > 0 }
8  if C mod K = 0 then { possible }
9  Q := C div K
10 else { impossible }
11 Q := -1
12 end
```


Snoep 10: Beter?

```
1    { solve (? Q : : K * Q = C) for K, C >= 0 }
2
3  ; if K = 0 then begin { special case }
4      if C = 0 then { possible }
5        Q := 0 { any value Q will do }
6      else { impossible }
7        Q := -1
8    end
9    else begin { K > 0 }
10     if C mod K = 0 then { possible }
11       Q := C div K
12     else { impossible }
13       Q := -1
14   end
```


Opmerkingen bij Practicum 1

- In Pascal: variabelen zijn initieel ongedefinieerd
- `/` is geen operator op `Integer` (maar op `Real`)
- Delen door 0 kan niet: **`div`**, **`mod`**, `/` run-time error
- Restricties op invoer hoef je niet te controleren
- PEACH controleert letterlijk: extra uitvoer, grote/kleine letters

Commentaar in (Pascal) programma's

{ ⟨*tekst zonder '}'*⟩ }
// ⟨*tekst tot regeleinde*⟩

- Aanhef: auteur, id.nr., datum, korte toelichting
- Variabele declaratie: rol toelichten
- Opdrachten: doel toelichten

  Pas op voor *overbodig* commentaar

```
i := i + 1 //i verhogen met 1
```


Commentaar versus Annotatie met asserties

```
1  { bepaal aantal
2    snoepjes per kind }
3  Q := C div K
```

```
1  {@ (0 < K)  and (C mod K = 0) }
2  Q := C div K
3  {@ K * Q = C }
```

- Commentaar in natuurlijke taal verwijst vaak vooruit
- Annotatie doet assertie over de toestand ter plekke: {@ ... }
- Assertie = predikaat op de toestandsruimte

VolleybalScore: Aanhef en Toestandsruimte

```
1  program VolleybalScore;
2    { (c) 2002-2008, Tom Verhoeff }
3    { Houd stand van set bij voor volleybal (oude regels).
4      De score gaat alleen omhoog (met 1) als scorende team
5      de service heeft (vorige keer ook scoorde). }
6
7    { Programma in "kaal" Pascal }
8
9  var
10   score1 : Integer; { score van team 1 (>=0, uitvoer) }
11   score2 : Integer; { score van team 2 (>=0, uitvoer) }
12   service: Integer; { 1 of 2, team dat de service heeft }
13   scoort : Integer; { 1 of 2, team dat nu scoort (invoer) }
```


VolleybalScore: Initialisatie

```
15 begin
16   score1  := 0
17 ; score2  := 0
18 ; service := 1
```

VolleybalScore: Herhaling met `while`

```
20 ; while ( ( score1 < 25 ) and ( score2 < 25 ) )
21      or ( abs ( score1 - score2 ) < 2 )
22   do begin
23     write ( 'Welk team scoort er (1 of 2)? ' )
24   ; readln ( scoort )
25   ; if scoort = service then begin
26       if scoort = 1 then
27         score1 := score1 + 1
28       else {@ scoort = 2 }
29         score2 := score2 + 1
30     end
31   ; writeln ( 'De score is nu: ', score1, ' - ', score2 )
32   ; service := scoort
33   end
```


VolleybalScore: Afsluiting

```
35   {@ ( (score1 >= 25) or (score2 >= 25) ) and
36      ( abs(score1 - score2) >= 2 ),
37      dus score1 <> score2 }
38
39 ; if score1 > score2 then
40     writeln ( 'Team 1 wint' )
41   else {@ score2 > score1 }
42     writeln ( 'Team 2 wint' )
43
44 end.
```


Meer Pascal

- Constante benoemen: **const** ⟨*constnaam*⟩ = ⟨*constexpr*⟩ ;
- Nieuw type construeren
  - Interval-type: ⟨*constexpr*⟩ .. ⟨*constexpr*⟩
- Type benoemen: **type** ⟨*typenaam*⟩ = ⟨*type*⟩ ;

Constanten een naam geven

const ⟨*constnaam*⟩ = ⟨*constexpr*⟩ ;

Voorbeelden:

```
const NSpelers = 5; { aantal spelers, NSpelers >= 1 }

const Pi = 355 / 113;

const
  EersteLetter = 'a';
  TweedeLetter = succ ( EersteLetter );
```

Waarom? Verhoogde leesbaarheid en onderhoudbaarheid

Voorgedefinieerde Data Types

(Data) type: waardenverzameling met bijbehorende operaties

`Integer`: `-1  0  1  MaxInt  +  div  abs  sqr`

`Char`: `'a'  '9'  ' '  '"'  ''''  succ  pred`

`Boolean`: `False  True  not  and  or`

`Real`: `0.0  -3.14  1e6  sqrt  exp  sin`

Interval-type

⟨*constexpr*⟩ `..` ⟨*constexpr*⟩

Erft operaties van "moedertype", mits van toepassing

Voorbeelden:

```
1 .. MaxInt { positieve gehele getallen }

'a' .. 'z' { de kleine letters van het Latijnse alfabet }
```

Waarom gebruiken? Automatische controle mogelijk

Types een naam geven

type ⟨*typenaam*⟩ = ⟨*type*⟩ ;

Voorbeelden:

```
type Speler = 1 .. NSpelers;

type KleineLetter = 'a' .. 'z';
```

Waarom? Verhoogde leesbaarheid en onderhoudbaarheid

Nette uitvoer produceren

write (E : M) $(M > 0)$

Drukt expressie E af met (minimum) veldbreedte M

Weergave van E wordt links aangevuld met zoveel spaties
als nodig om in totaal M karakters af te drukken

Weergave van E wordt niet ingekort als lengte $> M$

write (E : M : D) $(M > 0,\ D \geq 0)$

Drukt reële expressie E af met D decimalen (anders wetensch. not.)

VolleybalScore (2): Aanhef en definities

```
1  program VolleybalScore2;
2    { (c) 2002-2008, Tom Verhoeff, Versie 2 }
3    { Houd stand van set bij voor volleybal (oude regels).
4      De score gaat alleen omhoog (met 1) als het scorende team
5      de service heeft (vorige keer ook scoorde). }
6
7  const
8    MinScore     = 25; { minimum score nodig om te winnen, > 0 }
9    MinVerschil  =  2; { minimum verschil nodig om te winnen, > 0 }
10
11 type
12   Team         = 1 .. 2;
13   Score        = 0 .. MaxInt;
```


VolleybalScore (2): Toestandsruimte en initialisatie

```
15 var
16   score1 : Score; { score van team 1 (uitvoer) }
17   score2 : Score; { score van team 2 (uitvoer) }
18   service: Team;  { team dat de service heeft }
19   scoort : Team;  { team dat nu scoort (invoer) }
20   winnaar: Team;  { winnaar na afloop (uitvoer)}
21
22 begin
23   score1  := 0
24 ; score2  := 0
25 ; service := 1
```


VolleybalScore (2): Herhaling

```
27 ; while ( ( score1 < MinScore ) and ( score2 < MinScore ) )
28      or ( abs ( score1 - score2 ) < MinVerschil )
29   do begin
30     write ( 'Welk team scoort er (1 of 2)? ' )
31   ; readln ( scoort )
32   ; if scoort = service then begin
33       if scoort = 1 then
34         score1 := score1 + 1
35       else {@ scoort = 2 }
36         score2 := score2 + 1
37     end
38   ; writeln ( 'Stand: ', score1 : 3, ' - ', score2 : 3 )
39   ; service := scoort
40   end
```


VolleybalScore (2): Afsluiting

```
42   {@ ( (score1 >= MinScore) or (score2 >= MinScore) ) and
43      ( abs(score1 - score2) >= MinVerschil ),
44      dus score1 <> score2 }
45
46 ; if score1 > score2 then
47     winnaar := 1
48   else {@ score2 > score 1 }
49     winnaar := 2
50
51 ; writeln ( 'Team ', winnaar, ' wint' )
52 end.
```

Specificatie voor ronde van dobbelspel

Schrijf een programma dat de 5 worpen inleest, en de hoogste worp en de eventuele winnaar bepaalt.

Invoer De invoer wordt aangeboden op standaard invoer. De eerste (en enige) regel bevat 5 getallen (uit het interval 1..12) gescheiden door spaties.

Uitvoer De uitvoer moet geschreven worden naar standaard uitvoer. De eerste (en enige) regel bevat het grootste van de invoergetallen en het volgnummer van de winnaar (1..5) dan wel een nul (0) als er geen winnaar is. De twee getallen dienen gescheiden te worden door één spatie.

Ontwerp voor ronde van dobbelspel

1. Lees de worpen in en onthoud ze
2. Bepaal het maximum van de ingelezen worpen
3. Bepaal het aantal keer dat het maximum voorkomt
4. Bepaal wie dit maximum als eerste wierp
5. Bepaal de eventuele winnaar
6. Schrijf maximum en winnaar

Alternatief ontwerp

1. Lees de worpen één voor één in en houd ondertussen bij
   (a) wat het maximum *van de ingelezen worpen* is
   (b) wie de winnaar *van de ingelezen worpen* is
2. Schrijf maximum en winnaar

Dobbelen (2): Aanhef en definities

```
1  program Dobbelen2;
2    { (c) 2001, Tom Verhoeff, Versie 2 }
3    { Lees 5 worpen en bepaal de uitslag
4      (werper van het unieke maximum) }
5
6  const
7    NSpelers  =  5; { aantal spelers }
8    MaxWaarde = 12; { maximale waarde van een worp }
9
10 type
11   Speler = 1 .. NSpelers;  { de verzameling spelers }
12   Waarde = 1 .. MaxWaarde; { de verzameling worpwaarden }
```

Dobbelen (2): Toestandsruimte

```
13 var
14   worp: Waarde;
15     { worp van speler i (invoer) }
16
17   i: Speler;
18     { om worpen van ronde te doorlopen }
19
20   max: Waarde;
21     { maximum worp van spelers 1 t/m i }
22
23   winnaar: 0 .. NSpelers;
24     { winnaar van 1 t/m i; 0 = geen winnaar (uitvoer) }
```


PEACH

- Gedragscode Computergebruik TU/e: alle transacties gelogd
- Opgaven individueel oplossen en inzenden.
- Inzendingen worden automatisch onderling vergeleken.
- Inzendtermijn heeft harde deadline.
- Verlenging alleen bij tijdig en onderbouwd verzoek.
- Alleen definitieve versie inzenden; niet experimenteren.

Aandachtspunten bij programma's

Programma's beschouwen als (mooi) product!

- Functionele correctheid (volgens gegeven specificatie)
- Nette opmaak
- Helder commentaar
- Met behandelde middelen

Deze lijst wordt nog uitgebreid!

Extra aandachtspunten bij programma's

- Zinvol versus overbodig commentaar
- Commentaar versus annotatie
- Adequaat gebruik van constanten en types
- Zinvolle naamgeving
- Volgens gegeven ontwerp