PROGRAMMEREN 2012

Leo Meerts

leo.meerts @ science.ru.nl

kamer HG01.713

telefoon 365 3023

Ben Ruijl

benruyl @ gmail.com

Thomas Bronzwaer

t.bronzwaer @ gmail.com

Luc Hendriks

luc.hendriks @ gmail.com

VANDAAG

INLEIDING

PROGRAMMEREN	PROBLEEM OPLOSSEN ALGORITMEN PROGRAMMEERMETHODEN ALGEMENE TAALSTRUCTUREN PROGRAMMEERSTIJL DEBUGGEN NAUWKEURIGHEID EFFICIENTIE COMMENTAAR
C	BEGINSELEN PROGRAMMASTRUCTUUR
PRACTICUM	LINUX <-> MS-WINDOWS WENNEN AAN SYSTEEM EERSTE DRIE OPDRACHTEN

ALGEMEEN

PRACTICUM	MAANDAGMIDDAG NA COLLEGE
	OPDRACHTEN (ZIE TIJDSCHEMA) VOLDOENDE --> EINDCIJFER
	INDIVIDUEEL MAKEN
	UITERLIJK NA EEN WEEK INLEVEREN
	ASSISTENTIE DOOR ASSISTENTEN EN DOOR DOCENT
LITERATUUR	HANDLEIDING
	BOEK “The C Programming Language”

OPDRACHTEN MAKEN

INDIVIDUEEL WERKEN AAN OPDRACHTEN

HOUDT O.A. IN

- DAT ER ONDERLING OVER OPDRACHTEN EN HET OPLOSSEN DAARVAN GESPROKEN MAG WORDEN
- DAT ER RAAD GEVRAAGD EN GEGEVEN MAG WORDEN, OOK AAN/DOOR DOCENT EN ASSISTENTEN

HOUDT NIET IN

- DAT ER (AL DAN NIET MET MEDEWETEN) OPDRACHTEN OF DELEN DAARVAN VAN ANDEREN GEKOPIEERD WORDEN EN ALS EIGEN WERK INGELEVERD WORDEN

ALS DIT LAATSTE ONTDEKT WORDT, KUN JE IN IEDER GEVAL HET VAK VOOR DIT STUDIEJAAR VERGETEN

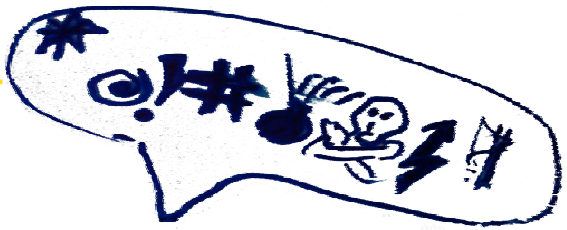
PROBLEEM OPLOSSEN

PROBLEEM OPLOSSEN

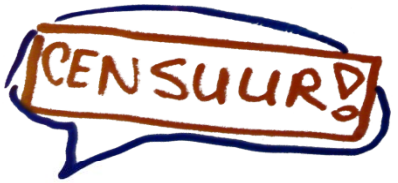
- EERST HET PROBLEEM BEGRIJPEN
- IS ER WEL EEN COMPUTER NODIG?
- RECEPT OPSTELLEN DAT PROBLEEM OPLOST
- RECEPT CONTROLEREN

IMPLEMENTATIE

- RECEPT OMZETTEN IN PROGRAMMEERTAAL
  KLAAR ! ! ALHOEWEL . .

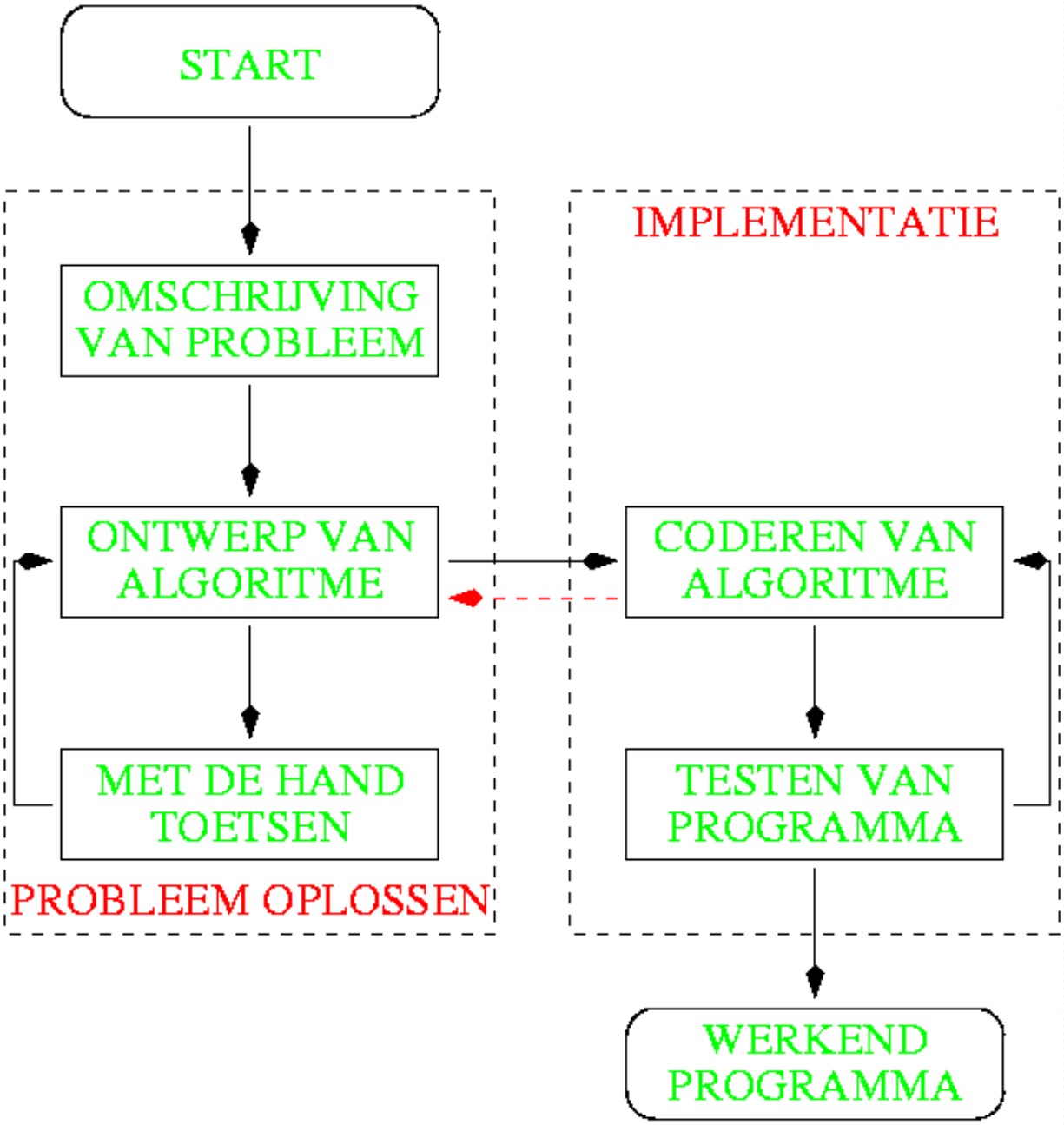


- PROGRAMMA COMPILEREN
  (TAALFOUTEN VERBETEREN ! ! !)



- PROGRAMMA TESTEN
  (LOGISCHE FOUTEN VERBETEREN ! ! !)

KLEIN PROBLEEM



VOORBEELD (1)

KOP KOFFIE UIT AUTOMAAT HALEN:

- ZORG VOOR VOLDOENDE MUNTEN
- WERP MUNTEN IN GLEUF
- MAAK KEUZE
- WACHT TOT BEKER GEVULD IS
- PAK BEKER
- DRUK OP KNOP VOOR WISSELGELD
- PAK WISSELGELD

VOORBEELD (2)

KOP KOFFIE UIT AUTOMAAT GEVEN:

- WACHT OP VOLDOENDE GELD
- WACHT OP KEUZE
- GEEF KOP KOFFIE
- ALS TERUGGAVEKNOP INGEDRUKT WORDT:
  GEEF WISSELGELD TERUG

VOORBEELD (3)

KOP KOFFIE UIT AUTOMAAT GEVEN:

```
ZOLANG    ER NOG KOFFIE IS
DOE

    ALS   GELDINJECTIE
    DAN   PAS TOTAALBEDRAG AAN

    ALS   KEUZEKNOP
    EN    TOTAALBEDRAG GROOT GENOEG
    DAN   GEEF KOP KOFFIE
          PAS TOTAALBEDRAG AAN

    ALS   TERUGGAVEKNOP
    EN    WISSELGELD AANWEZIG
    DAN   GEEF WISSELGELD TERUG
          PAS TOTAALBEDRAG AAN
```

KENMERKEN VAN ALGORITME

EEN ALGORITME BESTAAT UIT STAPPEN, DIE EEN PROCES BESCHRIJVEN
  EEN PROCESSOR MOET DIT ALGORITME KUNNEN BEGRIJPEN

EEN ALGORITME IS ALTIJD EINDIG, DE UITVOERING HOEFT NIET EINDIG TE ZIJN

EEN ALGORITME HEEFT EEN BEPAALD ABSTRACTIENIVEAU

ELKE STAP IS EENDUIDIG

SERIELE, COLLATERALE OF PARALLELLE UITVOERING

SAMENGESTELD VS. ELEMENTAIR ALGORITME

INVOER EN UITVOER BIJ ALGORITMEN

PARAMETERS, LOKALE OBJECTEN

SAMENGESTELDE VS. ELEMENTAIRE OBJECTEN

BEWERINGEN OVER ALGORITME

VOORWAARDEN

WANNEER WERKT HET ALGORITME

CORRECTHEID

WERKT HET ALGORITME FOUTLOOS

TERMINATIE

WANNEER STOPT HET ALGORITME

EFFICIENTIE

HOE EFFICIENT IS HET ALGORITME

TOP-DOWN METHODE

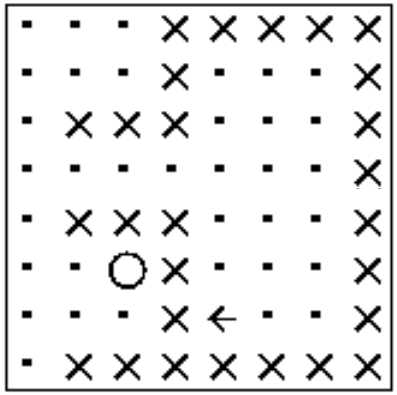
groot probleem opdelen in een aantal kleine, overzichtelijke problemen

voordeel: opzet is analoog aan algoritme

nadeel: bij complexe problemen zijn meestal meerdere iteraties nodig

VOORBEELD TOP-DOWN

Kareltje haalt zijn krant vanuit zijn bed



→ Kareltje met zijn neus naar rechts

× muur

· beweegruimte

○ krant

Kareltje haalt zijn krant vanuit zijn bed:
- ga van bed naar krant
- pak krant
- ga terug naar bed

VERFIJNINGEN

ga van bed naar krant:
- loop naar uitgang
- loop door uitgang
- loop naar krant

loop naar uitgang:
- rechtsom
- stap
- stap
- stap

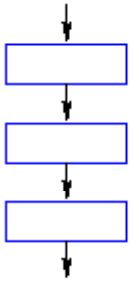
loop door uitgang:
- linksom
- stap
- stap
- stap
- stap

ELEMENTAIRE ALGORITMEN:
- rechtsom
- linksom
- stap
- pak krant
- leg krant neer

TAALSTRUCTUREN

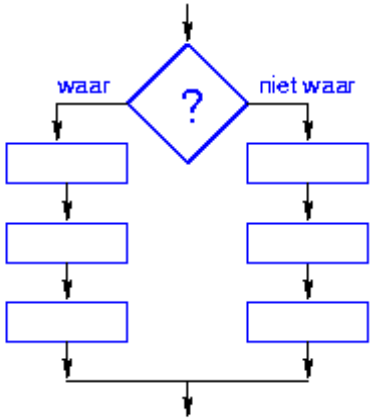
REEKS:

```
instructie 1
instructie 2
instructie 3
```



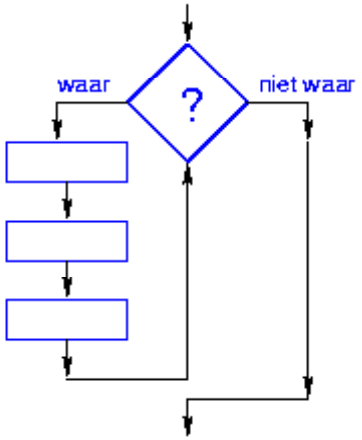
KEUZE:

```
ALS voorwaarde DAN
   instructie 1
   instructie 2
   instructie 3
ANDERS
   instructie 4
   instructie 5
   instructie 6
EINDE
```



HERHALING:

```
ZOLANG voorwaarde DOE
   instructie 1
   instructie 2
   instructie 3
EINDE
```



PROGRAMMEERSTIJL

BETER LEESBARE PROGRAMMA'S DOOR:

de STRUCTUUR van het programma duidelijk aan te geven:

- blanco regels tussenvoegen als afscheiding
- inspringen bij keuzen en herhalingen
- indeling in programmadelen aangeven

goed COMMENTAAR op de juiste plaats:

- omschrijving van de functie van het programma
- omschrijving van variabelen en constanten
- omschrijving van functie van programmadelen
- goed commentaar hoeft niet lang te zijn!

SPREKENDE NAMEN te gebruiken:

- dus geen a, b, i, ii, iii, enz.

VOORBEELD (1)

```
program sierdirt(input,output);{zeef van sierpinski}uses crt,graph;const
mag=80.00;p=5;pi=3.141593;shx=140.00;shy=200.00;var n,n1,l,m,xl,xr,xt,yb,yt,
GM,GD:integer;t,t1,t2,t3,t4,x,y,e2:real;begin gd:=detect;Initgraph(gd,GM,'');
outtextxy(24,16,'     Zeef van Sierpinski');t1:=pi/6.0;t2:=ln(2);t4:=sqrt(3);
for n:=0 TO p do begin for m:=0 to round(exp(ln(3)*n))-1 do begin x:=0;y:=0;l
:=m;for n1:=0 TO n-1 do begin t:=((l mod 3)*4+1)*t1;t3:=exp(t2*n1);x:=x+COS(t)
/t3;y:=y+sin(t)/t3;l:=l div 3;end;e2:=exp(t2*(n+1));xr:=round((x+t4/e2)*mag+
shx);xl:=round((x-t4/e2)*mag+shx);yb:=round((y-1.0/e2)*mag+shy);yt:=round((y+
1.0/exp(t2*n))*mag+shy);xt:=round(x*mag+shx);line(xr,yb,xt,yt);line(xt,yt,xl
,yb);line(xl,yb,xr,yb);end;end;outtextxy (24,300,'Geef RETURN.');READLN;
restorecrtmode;end.
```

VOORBEELD (2)

```
PROGRAM SIER (INPUT, OUTPUT);

{============================================================
=
{ file:       SIER.PAS
{ status:     03-jul-1990, P.R.O. Gram (v13.6)
{ funktie:    Teken een "Zeef van Sierpinski".
{
{ De figuur is opgebouwd uit een aantal nivo's van steeds kleiner wordende
{ gelijkzijdige driehoeken:
{   nivo 0 bevat 1 driehoek,
{   nivo 1 bevat 3 driehoeken,
{   nivo 2 bevat 3*3 driehoeken,
{   nivo 3 bevat 3*3*3 driehoeken, enzovoort.
{
{ Per nivo worden voor elke tot dat nivo behorende driehoek de coordinaten
{ van het centrum berekend en hieruit de hoekpuntcoordinaten (deze worden
{ als aanpassing aan de schermcoordinaten met een vergrotingsfaktor
{ vermenigvuldigd en met een waarde in x resp y vermeerderd), waarna de
{ betreffende driehoek wordt getekend.
{
{ Schermcoordinaten:
{   de x loopt van 0 tot max_x (van links naar rechts),
{   de y loopt van 0 tot max_y (van boven naar onder).
{============================================================}

USES crt, dos, graph;

CONST
  c_magx   = 0.667;    { vergrotingsfaktor voor figuur in X }
  c_magy   = 1.000;    { vergrotingsfaktor voor figuur in Y }
  c_shx    = 1.00;     { verschuiving voor figuur in X }
  c_shy    = 1.20;     { verschuiving voor figuur in Y }
  max_nivo = 5;        { maximum aantal nivo's (excl nivo 0) }

VAR
  gdriv,               { type van grafische driver }
  gmode,                 { grafische mode }
  max_x,
  max_y,
  nivo,                { nivo waarop driehoeken getekend worden }
  subcnt,               { tijdelijke teller }
  tmpint,               { tijdelijke variable }
  triang,              { nummer van driehoek binnen een nivo }
  ErrorCode
  : INTEGER;
  angle,               { hoek tussen pos x-as en centrum driehoek }
  magx,
  magy,
  shx,
  shy,
  tmpvar,               { tijdelijke variabele }
  x,                   { x-coordinaat van centrum driehoek }
  y,                   { y-coordinaat van centrum driehoek }
  xl,                  { x-coordinaat van linker hoek driehoek }
  xr,                  { x-coordinaat van rechter hoek driehoek }
  xt,                  { x-coordinaat van top driehoek }
  yb,                   { y-coordinaat van basis driehoek }
  yt                   { y-coordinaat van top driehoek }
  : REAL;
  uur_v, min_v, sec_v, csec_v,
  uur_n, min_n, sec_n, csec_n
  : WORD;
  uur1, uur2,
  min1, min2,
  sec1, sec2,
  csec1, csec2,
  tijd
  : REAL;
  tijd_string          { bevat de berekende tekentijd }
  : STRING;

BEGIN

{ Initializatie. }

  { maak het beeldscherm schoon }
  CLRSCR;

  { zoek uit welke grafische kaart er gebruikt wordt en probeer deze te
    initializeren (ga in grafische mode als alles goed gaat) }
  gdriv := DETECT;                    { kijk welke grafische kaart }
  INITGRAPH (gdriv, gmode, '');       { laad driver voor aansturing }

  { test of initializatie van grafische kaart goed gegaan is }
  IF GRAPHRESULT <> GROK THEN
  BEGIN                               { fout: schrijf boodschap en stop }
    WRITELN ('Fout bij initializeren van grafische kaart:');
    WRITELN ('  grafische kaart = ', gdriv);
    WRITELN ('  grafische mode  = ', gmode);
    WRITELN;
    WRITELN ('Geef RETURN om door te gaan.');
    READLN;
    HALT;                             { stop het programma }
  END;

  { vraag maximale X-waarde voor scherm,
    bereken hieruit verschuiving en vergroting in X-richting }
  max_x := GetMaxX;
  shx  := c_shx * (max_x + 1) / 2.00;
  magx  := c_magx * (max_x + 1) / 4.00;

  { vraag maximale Y-waarde voor scherm,
    bereken hieruit verschuiving en vergroting in Y-richting }
  max_y := GetMaxY;
  shy   := c_shy * (max_y + 1) / 2.00;
  magy  := c_magy * (max_y + 1) / 4.00;

  { titel van programma op scherm }
  OutTextXY (max_x DIV 16, max_y DIV 10,'      Zeef van Sierpinski');

{ Teken de figuur. }

  { neem begintijd op }
  GetTime (uur_v, min_v, sec_v, csec_v);

  { teken de driehoeken nivo voor nivo }
  FOR nivo:=0 TO max_nivo DO
  BEGIN

    { teken alle driehoeken voor een nivo }
    { de uitdrukking "EXP(LN(n)*m)" berekent "n tot de macht m" }
    FOR triang:=0 TO ROUND(EXP(LN(3)*nivo))-1 DO
    BEGIN
      x := 0;  y := 0;
      tmpint := triang;

      { bereken de centrumcoordinaten van de huidige driehoek }
      { deze coordinaten worden berekend t.o.v. de oorsprong (0,0) }
      FOR subcnt:=0 TO nivo-1 DO
      BEGIN
        angle := ((tmpint MOD 3) * 4 + 1) *  pi / 6.0;
        tmpvar := EXP(LN(2)*subcnt);
        x := x + COS(angle)/tmpvar;
        y := y + SIN(angle)/tmpvar;
        tmpint := tmpint DIV 3;
      END;

      { bereken de hoekcoordinaten van de huidige driehoek }
      { deze coordinaten worden berekend als schermcoordinaten }
      xr := (x + SQRT(3) / EXP(LN(2)*(nivo+1))) * magx + shx;
      xl := (x - SQRT(3) / EXP(LN(2)*(nivo+1))) * magx + shx;
      yb := (y - 1.0     / EXP(LN(2)*(nivo+1))) * magy + shy;
      yt := (y + 1.0     / EXP(LN(2)* nivo   )) * magy + shy;
      xt :=  x                                  * magx + shx;

      { teken de huidige driehoek }
      line (ROUND(xr), ROUND(yb), ROUND(xt), ROUND(yt));
      line (ROUND(xt), ROUND(yt), ROUND(xl), ROUND(yb));
      line (ROUND(xl), ROUND(yb), ROUND(xr), ROUND(yb));

    END;                    { einde van loop over driehoeken }
  END;                      { einde van loop over nivo's }

  { neem eindtijd op }
  GetTime (uur_n, min_n, sec_n, csec_n);

  { bereken de benodigde tekentijd en druk deze af }
  uur1  := uur_v;   uur2  := uur_n;    { zet tijd om naar REAL }
  min1  := min_v;   min2  := min_n;
  sec1  := sec_v;   sec2  := sec_n;
  csec1 := csec_v;  csec2 := csec_n;
  tijd  := 3600.00 * (uur2 - uur1) + 60.00 * (min2 - min1) + sec2 - sec1 +
           (csec2 - csec1) / 100.00;   { tijd in sekonden }
  Str (tijd:10:2, tijd_string);
  OutTextXY (max_x DIV 20, max_y - max_y DIV 15,
             'Benodigde tijd: ' + tijd_string + ' sekonden.');

{ Einde van het programma. }

  OutTextXY (max_x DIV 20, max_y - max_y DIV 30,
             'Geef RETURN om door te gaan.');
  READLN;

  RestoreCRTMode;

END.
```

DEBUGGEN

Nalopen van de programmacode:

- logica van het programma
- wordt dezelfde variabele tegelijkertijd voor verschillende doeleinden gebruikt?
- kloppen de "randvoorwaarden" wel bij herhalingen?

Afdrukken van tussenresultaten:

- eerst denken, dan doen!
- programma nalopen aan de hand van de afgedrukte tussenresultaten

Debugger gebruiken:

- break points zetten
- stap–voor–stap door het programma gaan
- de waarde van variabelen bekijken en/of veranderen

NAUWKEURIGHEID

DE DIGITALE COMPUTER WERKT BINAIR

de waarde van de breuk 1/3 kan intern dus niet exact weergegeven worden

er treden daarom "afrondingsfoutjes" op

getallen met een decimale punt worden niet in alle gevallen exact weergegeven

gehele getallen worden wel exact weergegeven

OOK EEN COMPUTER HEEFT GRENZEN

er zijn grenzen aan het bereik van gehele en decimale getallen op een computer

hierdoor kunnen (grote) afwijkingen ontstaan

DE ENE COMPUTER IS DE ANDERE NIET

de interne weergave van decimale getallen is voor verschillende typen computers niet altijd hetzelfde

hetzelfde programma kan op verschillende computers verschillende resultaten geven!

CHAR_BIT	8	bits in een char
CHAR_MAX	UCHAR_MAX *of* SCHAR_MAX	maximum waarde van char
CHAR_MIN	0 *of* SCHAR_MIN	minimum waarde van char
INT_MAX	+32767	maximum waarde van int
INT_MIN	-32767	minimum waarde van int
LONG_MAX	+2147483647L	maximum waarde van long
LONG_MIN	-2147483647L	minimum waarde van long
SCHAR_MAX	+127	maximum waarde van signed char
SCHAR_MIN	-127	minimum waarde van signed char
SHRT_MAX	+32767	maximum waarde van short
SHRT_MIN	-32767	minimum waarde van short
UCHAR_MAX	255U	maximum waarde van unsigned char
UINT_MAX	65535U	maximum waarde van unsigned int
ULONG_MAX	4294967295UL	maximum waarde van unsigned long
USHRT_MAX	65535U	maximum waarde van unsigned short

De namen in de nu volgende tabel, een deelverzameling van de verzameling namen in <float.h>, zijn constanten die verband houden met rekenen met floating point-getallen. De getallen in de tabel zijn minimumwaarden voor de bijbehorende constanten. Welke waarde die constanten in werkelijkheid hebben, hangt van de implementatie af.

FLT_RADIX	2	grondtal bij exponent-weergave, bijvoorbeeld 2, 16
FLT_ROUNDS		floating point-afrondings*mode* voor optellen
FLT_DIG	6	aantal decimaalcijfers bij precisie
FLT_EPSILON	1E-5	kleinste getal x waarvoor $1.0 + x \neq 1.0$
FLT_MANT_DIG		aantal cijfers in mantisse bij grondtal FLT_RADIX
FLT_MAX	1E+37	grootste floating point-getal
FLT_MAX_EXP		grootste n waarvoor de waarde van FLT_RADIX^n–1 nog kan worden gerepresenteerd
FLT_MIN	1E-37	kleinste genormaliseerde floating point-getal
FLT_MIN_EXP		kleinste n waarvoor 10^n een genormaliseerd getal is
DBL_DIG	10	aantal decimale cijfers van precisie
DBL_EPSILON	1E-9	kleinste getal x waarvoor $1.0 + x \neq 1.0$
DBL_MANT_DIG		aantal cijfers in mantisse bij grondtal FLT_RADIX
DBL_MAX	1E+37	grootste floating-point getal van type double
DBL_MAX_EXP		grootste n waarvoor de waarde van FLT_RADIX^n–1 nog kan worden gerepresenteerd
DBL_MIN	1E-37	kleinste genormaliseerde float van type double
DBL_MIN_EXP		kleinste n waarvoor 10^n een genormaliseerd getal is

E:258

EFFICIENTIE

Een programma van een paar milliseconden dat tweemaal zo snel gemaakt wordt, levert niet veel absolute tijdwinst op.

Een programma van een paar uur wel!

Probeer een "slim" algoritme te maken

- eerst denken, dan doen!
- van anderen kan je leren

Herhalingen zijn belangrijk

- gooi extra ballast uit een herhaling
- breek een herhaling zo snel mogelijk af
- bereken vaak gebruikte uitdrukkingen vooraf

Buiten controle van de programmeur

- moderne compilers zijn goed in optimaliseren

COMMENTAAR (1)

COMMENTAAR

IN DE KODE VAN EEN PROGRAMMA

HEEFT ABSOLUUT

GEEN

INVLOED OP DE

SNELHEID

WAARMEE DAT PROGRAMMA

UITGEVOERD WORDT

! ! !

COMMENTAAR (2)

COMMENTAAR

IN DE CODE VAN EEN PROGRAMMA

HEEFT ABSOLUUT

WEL

INVLOED OP DE

BEOORDELING

VAN DAT PROGRAMMA

! ! !

! ! !

REKENKUNDIGE UITDRUKKINGEN

WISKUNDIGE NOTATIE

$3x + y$

C NOTATIE

```
3*x + y
```

WISKUNDIGE NOTATIE

$$\frac{3x + y}{a + b}$$

C NOTATIE

```
(3*x + y) / (a + b)
```

Hoe maak je een werkend programma

- Schrijf eerst je programma in C-code
- Compileer het programma
  - C-code wordt omgezet naar voor de computer leesbare (binaire) code
  - .o (object) file(s) worden gecreerd
- Link het programma
  - Bibliotheken worden bijgevoegd
  - .exe (executable) file wordt gemaakt

Voor de hele procedure kun je een programmeer werkomgeving gebruiken: Code::Blocks

Tips

- Denk na voordat je begint
- Probeer het wiel niet telkens zelf uit te vinden
- Maak gebruik van de voorbeelden
- Kijk of hierin code zit die je voor je eigen programma kunt gebruiken

Hello World

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  printf ("Hello World\n");

#ifdef _WIN32
  /* Necessary to prevent closing of window under Windows */
  system ("PAUSE");
#endif

  return (0);
}
```

Input Output

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  /* declarations */
  float floatOne, floatTwo, floatThree;

  /* message for user */
  printf ("Geef drie floating point getallen, gescheiden door een spatie:\n");

  /* read three floating point values from standard input */
  scanf ("%f %f %f", &floatOne, &floatTwo, &floatThree);

  /* print the three values on standard output */
  printf ("Ingevoerde waarden: %f, %f, %f\n", floatOne, floatTwo, floatThree);

  return (0);
}
```

Input Output Standard error

```
int main ()
{
  /* declarations */
  float floatOne, floatTwo, floatThree;

  /* message for user */
  printf ("Geef drie floating point getallen, gescheiden door een spatie:\n");

  /* read three floating point values from standard input */
  scanf ("%f %f %f", &floatOne, &floatTwo, &floatThree);

/* print the three values naar standard ERROR output */
  fprintf (stderr,"\nIngevoerde waarden geprint naar stderr: %f, %f, %f\n\n", floatOne, floatTwo,
     floatThree);

  /* print the three values on standard output */
  printf ("Ingevoerde waarden: %f, %f, %f\n", floatOne, floatTwo, floatThree);

  return (0);
}
```

fprintf ()

```
fprintf (stderr,
        "\nIngevoerde waarden geprint naar stderr: %f, %f, %f\n\n",
        floatOne, floatTwo, floatThree);
```

Nauwkeurigheid

Voorbeeld van (gebrek aan) nauwkeurigheid:
Dezelfde berekeningen worden uitgevoerd in
floating point en in double precision,
uitgaande van X = 0.000000000001 = 1.000000e-012.
Resultaten worden exponentieel (resp. float) weergegeven.
De exacte uitkomst is 1.0 (een)!

Berekende waarde	Resultaat in floating point	Resultaat in double precision
X^1	1.000000e-012	1.000000e-012
X^2	1.000000e-024	1.000000e-024
X^3	1.000000e-036	1.000000e-036
X^4	0.000000e+000	1.000000e-048
X^4 / X^1	0.000000e+000 (0.0)	1.000000e-036 (0.0)
X^4 / X^2	0.000000e+000 (0.0)	1.000000e-024 (0.0)
X^4 / X^3	0.000000e+000 (0.0)	1.000000e-012 (0.0)
X^4 / X^4	0.000000e+000 (0.0)	1.000000e+000 (1.0)