EE1400: Programmeren in C

BSc. EE, 1e jaar, 2012-2013, 1e college

Arjan van Genderen, Computer Engineering

13-11-2012

Overzicht van het vak

- Gegeven in Q2, week 2.1 – 2.6.
- Hoorcollege in week 2.1, 2.2, 2.4 en 2.5
- Elke week practicum
- Daarnaast ook zelfstudie en oefening vereist !
- Voortgangstoetsen in week 2.2 en 2.6
- Tentamen aan het einde van Q2 + herkansing einde Q3
  - Boek en collegeslides mogen erbij gehouden worden
- Docent: Arjan van Genderen
  - E-mail: A.J.vanGenderen@TUDelft.nl
  - Kamer: HB 16.270 / DW 1.070, Tel: 015 27 86217 / 82112
- Studiemateriaal:
  - Collegeslides
  - A Book on C, Al Kelly, Ira Pohl, Pearson Education, ISBN 0-201-18399-4 (verkrijgbaar bij ETV)
  - EE1400 Practicumhandleiding (Blackboard en TU dicatenverkoop)

Practicum (1)

- Grootste onderdeel van dit vak
- Wanneer en waar:
  - Ingeroosterde uren per week: 2/2/4/2/2/4, di. en do. middag:
    Gebouw Drebbelweg: student-assistenten beschikbaar.
    Voor details, zie groepsrooster op Blackboard.
  - Uren in eigen tijd: ongeveer 3/3/6/3/3/6
- Literatuur:
  - EE1400 Practicumhandleiding (Blackboard en TU dictatenverkoop)
- Software:
  - Code::Blocks onder Windows, Mac OS X of Linux (beschikbaar op practicum PCs maar ook op eigen PC simpel en gratis te installeren, zie Blackboard)

Practicum (2)

- Voor 10 opgaven (zie practicumhandleiding) moet je de C-code inleveren.
- Inleveren door uploaden op https://cpm.ewi.tudelft.nl
- Tijdens ingeroosterde practicum-uren student-assistenten beschikbaar.
- Bereid je goed voor en probeer van te voren al aan de opgaven te werken!
- Sowieso zul je ook in eigen tijd aan de opdrachten moeten werken.
- Oefenen is heel belangrijk
  - Overleggen mag
  - Code overnemen niet !

  Jij bent degene die het moet leren en …
  op het tentamen moet je het ook alleen doen.
- Bij onduidelijkheden: vraag assistenten, zoek in boek, zoek op internet, overleg met mede-studenten.
- Deadline opgaven hoofdstuk 1 t/m 3: 2 dec. 24.00h
- Deadline opgaven hoofdstuk 4 en 5: 28 dec. 24.00h

Hoorcollege 1

- Wat is programmeren ?
- Waarom C ?
- “Hello World!”: een simpel programma compileren en draaien
- Integrated Development Environment: Code::Blocks
- Kort overzicht van C
- Lexical elements/Syntax
- Operators
- Standard libraries

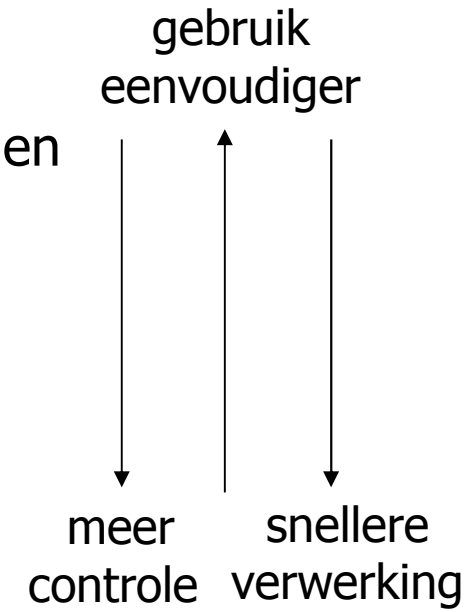
Corresponderende stof in boek: Hoofdstuk 0, 1 (t/m 1.6) en 2

Wat is Programmeren ?

"Het in een kunstmatige, geschreven taal vertellen wat de computer moet doen"

Verschillende manieren om computer te besturen:

- Een bestaand programma opstarten en met muis en toetsenbord besturen
- Scripttaal (Matlab, Perl, Tcl/Tk, PHP)
- Hogere programmeertaal (C, Java)
- Assembler (machine instructies)



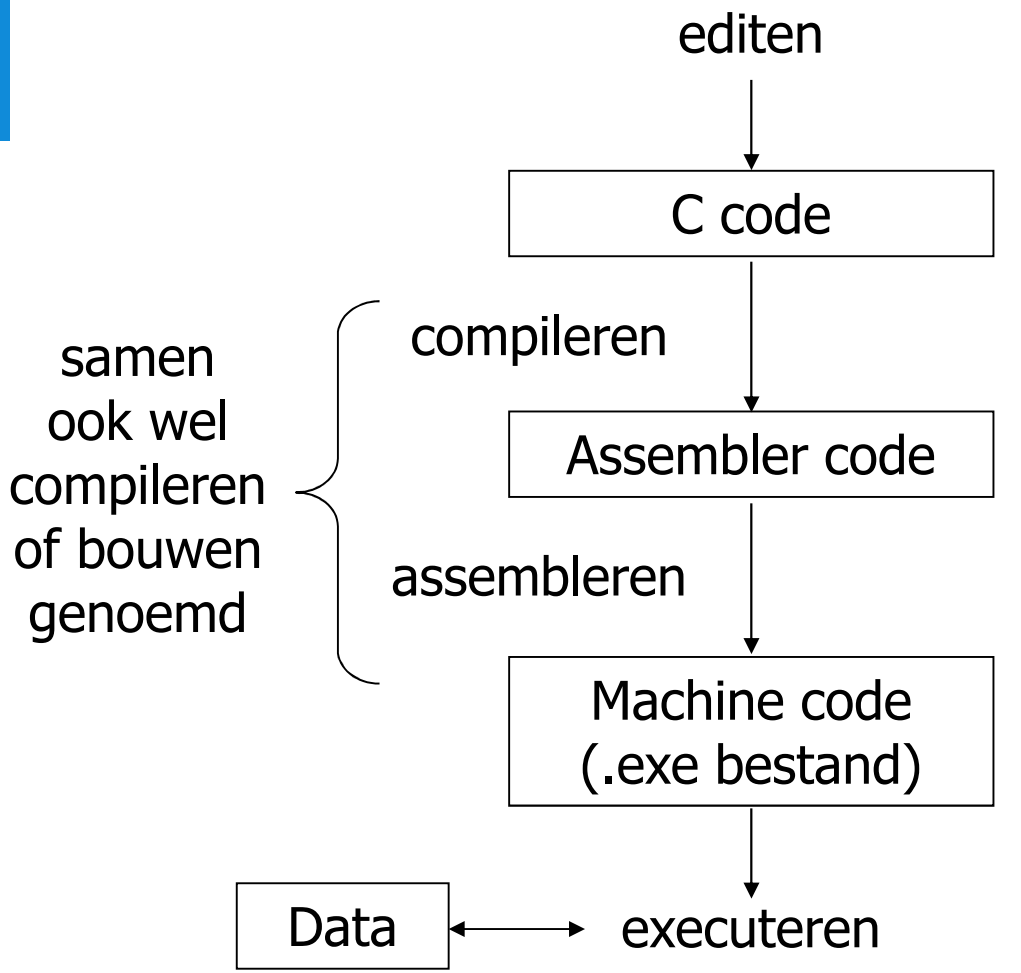
Een C programma

```
#include <stdio.h>

int main(void)
{
   int a;

   a = 10;
   printf("a is nu %d\n", a);

   a = a + 15;
   printf("a is nu %d\n", a);

   return 0;
}
```

Output:

```
a is nu 10
a is nu 25
```

Van C code tot programma uitvoeren (1)



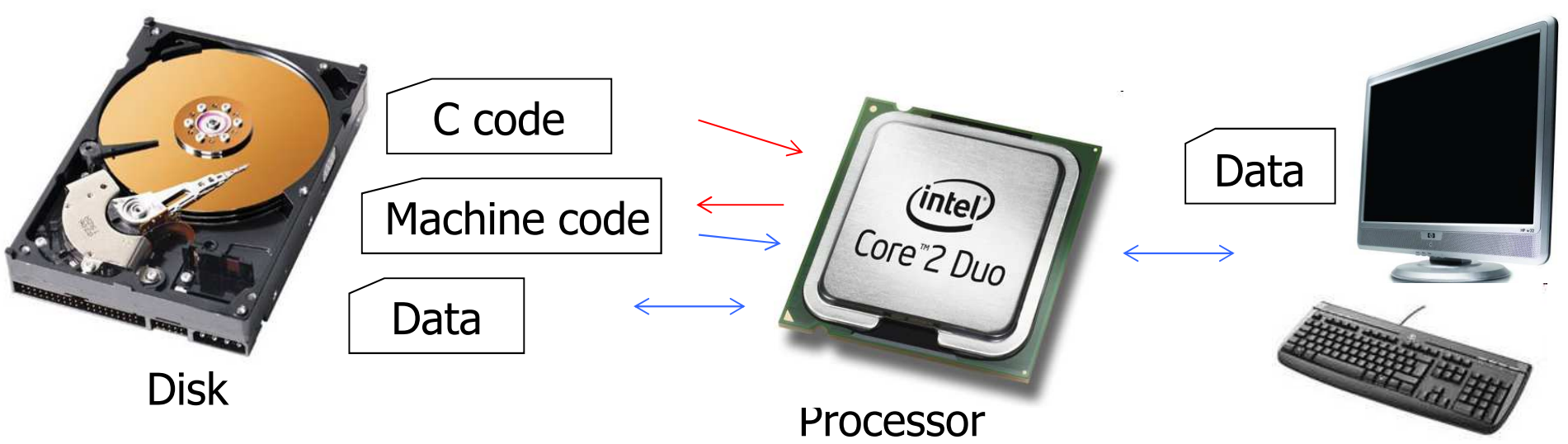
Voorbeeld:

a = a + 15;

```
load    R2
add     15
store   R2

0100  00000010
0110  00001111
0101  00000010
```

noodzakelijk maar
niet interessant

Van C code to programma uitvoeren (2)



Compileren

Executeren

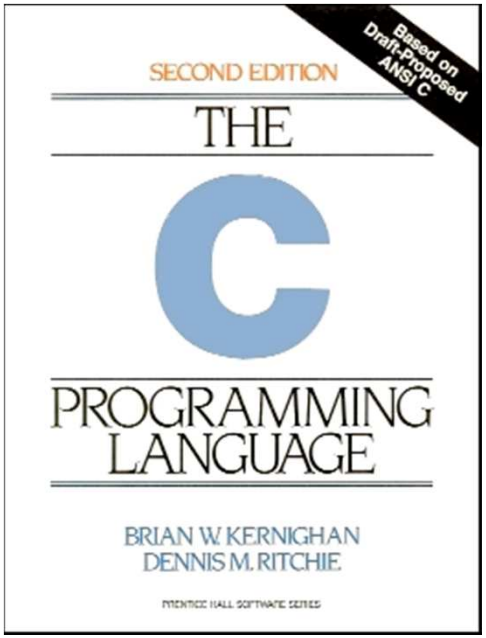
Wat is Programmeren nog meer ?

- Logisch nadenken
- Creatief zijn
- Secuur zijn: 1 karakter verkeerd en het programma kan onbruikbaar zijn: In 1996 blies de Ariadne 5 zichzelf op na de lancering door een programmeerfout, schade $ 500 miljoen
  - Testen is belangrijk
  - Niemand schrijft in 1 keer een groot foutloos programma: proberen en opnieuw proberen
- Verdeel en heers: breng hierarchie aan en hergebruik code:
  - Kies een goede opdeling in functies
  - Gebruik de juiste bibliotheken (veel is al beschikbaar !)

Waarom C ?

- C is de taal waarin (de kern van) Unix, Linux, Windows, Mac OS X, iOS en Android is geschreven
- C is een compacte taal
- C is “portable” (dezelfde C code draait op vele machines)
- C is efficient/krachtig
- C is modulair
- C is de basis van C++ en Java
- C “praat” makkelijk met hardware
- C wordt veel gebruikt (o.a. in EE)

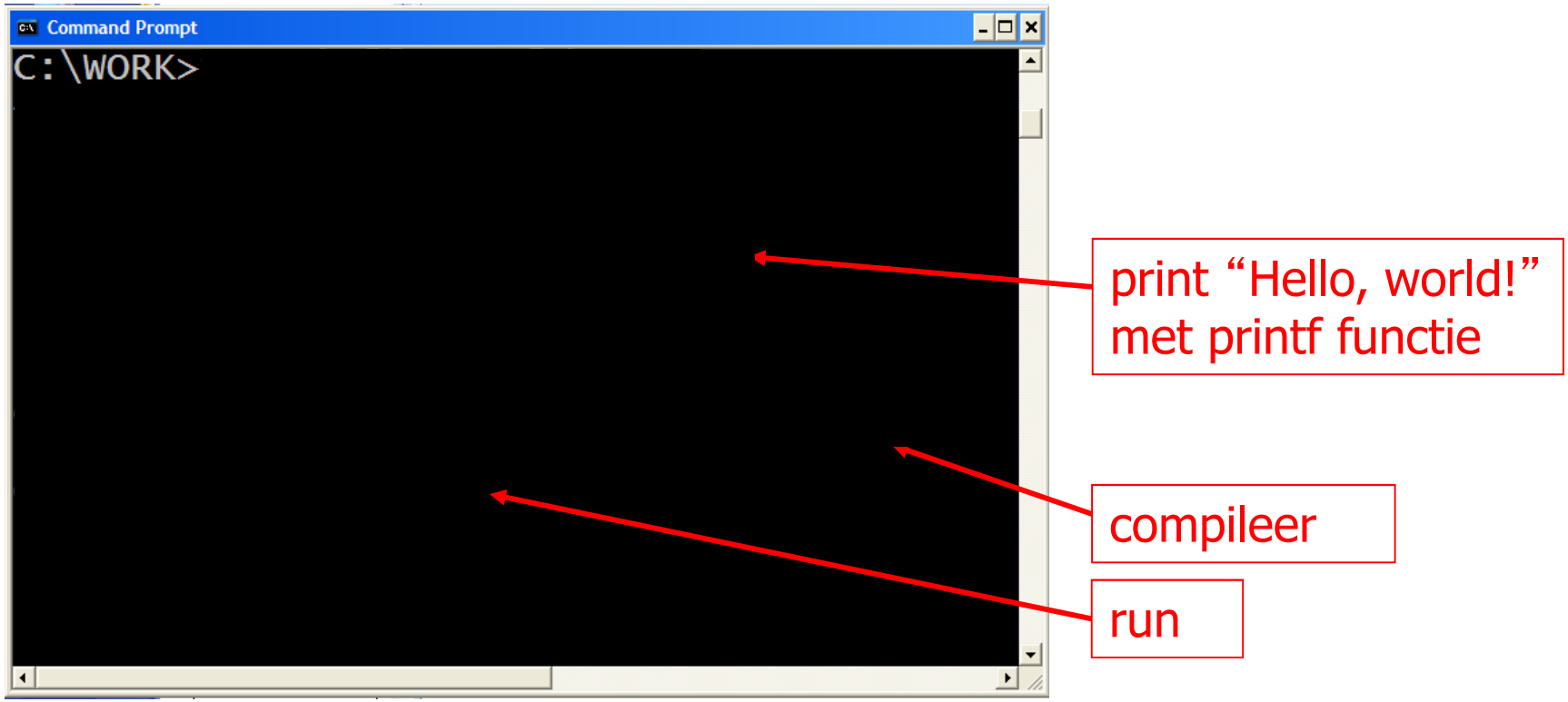
C wordt gebruikt bij EPO-2 !



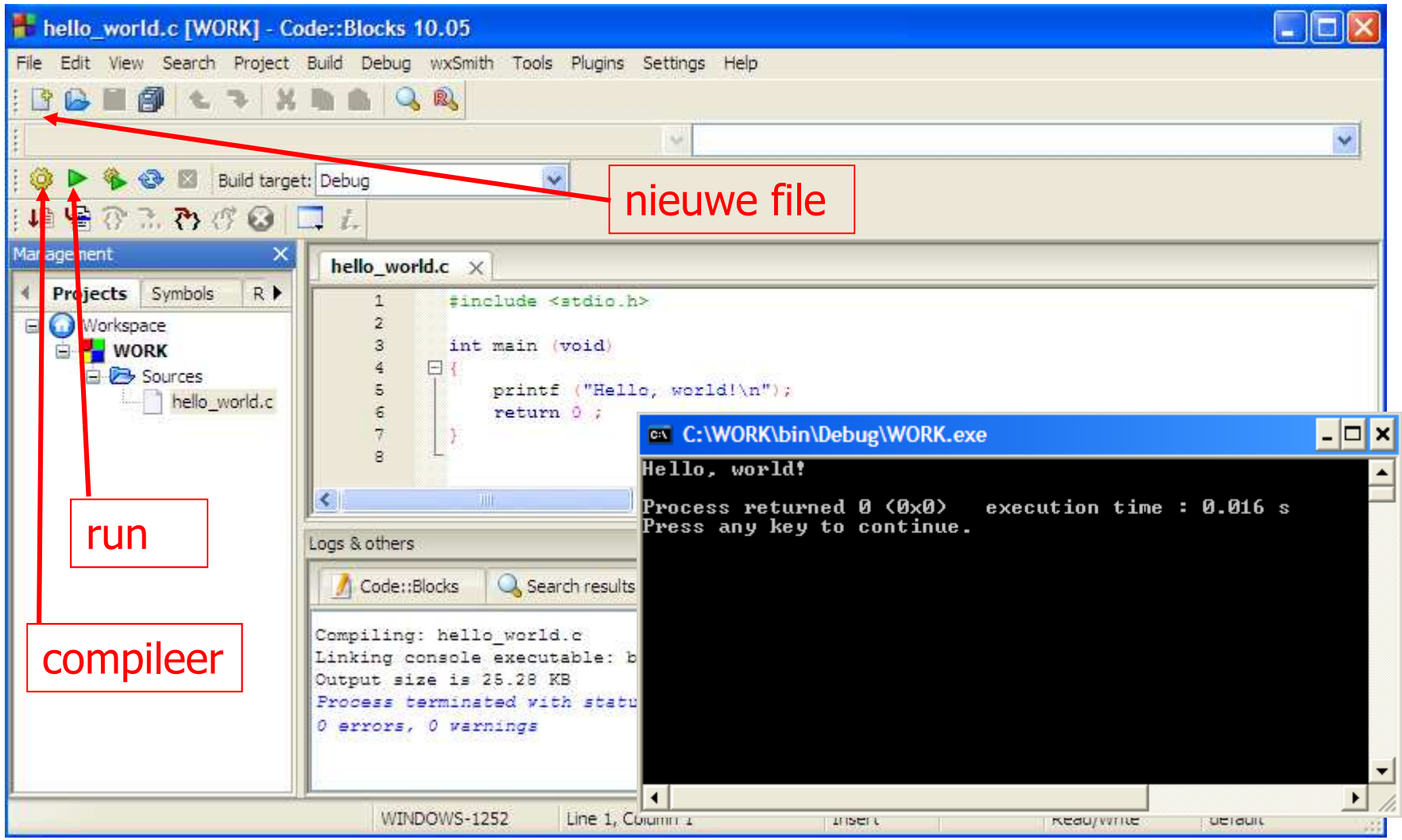
C is tussen 1969 en 1973 ontwikkeld door Dennis Ritchie bij Bell Telephone Laboratories.

“Hello, world!”

Het compileren en runnen van een eenvoudig programma



Integrated Design Environment: Code::Blocks



Kort Overzicht van C

Variabelen, Expressies en Assignments

```
/* The distance of a marathon in kilometers. */

#include <stdio.h>

int main(void)
{
   int     miles, yards;
   float   kilometers;

   miles = 26;
   yards = 385;
   kilometers = 1.609 * (miles + yards / 1760.0);
   printf("\nA marathon is %f kilometers.\n\n", kilometers);
   return 0;
}
```

commentaar

variabelen van het type int (“gehele getallen”)

float (“reëel getal”)

expressie

assignment

conversie karakter voor printen als floating point

Output:

```
A marathon is 42.185970 kilometers
```

#define en #include

- Voor compileren wordt altijd eerst automatisch de C preprocessor gedraaid
- Alle regels beginnend met # bevatten aanwijzingen voor de C preprocessor

```
#define  PI  3.14159
```
Vervang overal hierna PI door 3.14159

```
#include <stdio.h>
```
Neem inhoud van file stdio.h op hier (deze file staat op standaard plaats in het systeem)

```
#include "my_file.h"
```
Neem inhoud van eigen file my_file.h op hier

Printf en scanf

```
#include <stdio.h>

int main(void)
{
   char     c1, c2, c3;
   int      i;
   float    x;
   double   y;

   printf("\n%s\n%s",
      "Input three characters,",
      "an int, a float, and a double:  ");
   scanf("%c%c%c%d%f%lf", &c1, &c2, &c3, &i, &x, &y);
   printf("\nHere is the data that you typed in:\n");
   printf("%3c%3c%3c%5d%17e%17e\n\n", c1, c2, c3, i, x, y);
   return 0;
}
```

type char kan karakters volgens ASCII codering bevatten

double heeft meer precisie dan float

lees waarden in met scanf functie, let op & symbool

gereserveerde karakterposities in uitvoer

Input en output:

```
Input three characters
an int, a float, and a double: ABC  3  55  77.7
Here is the data that you typed in:
  A  B  C    3     5.500000e+01      7.770000e+01
```

Programmabesturing (1)

```
if (expr)
   statement
```

```
a = 1;
if (b == 3)
   a = 5;
printf("%d\n", a);
```

uitgevoerde stappen:

wanneer b is 3:

```
a = 1;
a = 5;
printf("%d\n", a);
```

wanneer b is geen 3:

```
a = 1;
printf("%d\n", a);
```

```
if (expr)
   statement1
else
   statement2
```

```
if (cnt == 0) {
   a = 2;
   b = 3;
   c = 5;
}
else {
   a = -1;
   b = -2;
   c = -3;
}
```

tussen haakjes: compound statement

wanneer cnt is 0:

```
a = 2;
b = 3;
c = 5;
```

wanneer cnt is geen 0:

```
a = -1;
b = -2;
c = -3;
```

Programmabesturing (2)

```
while (expr)
   statement
```

```
sum = sum + i;
```

```
i = i + 1;
```

```
i = 1;
while (i <= 3) {
   sum += i;
   ++i;
}
```

uitgevoerde stappen:

```
i = 1;
```
i is 1
```
sum += i;
++i;
```
i is 2
```
sum += i;
++i;
```
i is 3
```
sum += i;
++i;
```
i is 4 einde while lus

```
for (expr1; expr2; expr3)
   statement
```

```
expr1;
while (expr2){
   statement
   expr3;
}
```

```
for (i = 1; i <= 3; ++i) {
   sum += i;
}
```

Lexical elements/Syntax

Het compilatie proces

Tokens:

- keywords
- identifiers
- constants
- string constants
- operators
- punctuators

Voorbeelden:

```
for  while  int  float  return
maxValue   john   c1  x_12
5    120.27  'a'
"Hello World!\n"
+   -   *  /  %
;  ,  {  [
```

Identifiers

- Identifiers worden gebruikt voor namen van o.a. variabelen en functies
- Een identifier bestaat uit een letter of underscore, gevolgd door 0 of meer letters, underscores en/of cijfers

identifiers	geen identifiers
`k`	`not#me`
`_id`	`101_south`
`iamanidentifier2`	`-plus`
`so_am_i`	

- Keywords kunnen geen identifier zijn
- Kies zinvolle namen !

Commentaar

```
/* a comment */   /*** another comment ***/   /******/
```

- Alle commentaar (tussen /* en */) wordt door de compiler genegeerd
- Voor de programmeur (en zijn lezers) kan het heel waardevol zijn. Bijvoorbeeld:
  - Toelichting variabelen, functies
  - Uitleg algoritmen

```
/*
 * A common comment can be written in this fashion
 * to set it off from the surrounding code
 */

/*****************************
 *  If you wish, you can     *
 *  put comments in a box.   *
 *****************************/
```

Operators

Arithmetische operators

Arithmetische operators:

```
+    -    *    /    %
```

Operators worden gebruikt om expressies te maken:

```
a = 17 * (b + c);
```

De operator `/` voor integer operanden geeft delen met afkappen:

`5 / 3` geeft waarde `1`

De operator `%` geeft rest

`5 % 3` geeft waarde `2`

Increment en decrement

```
++i
i++
```
} verhoog i met 1 (increment)

```
i--
--i
```
} verlaag i met 1 (decrement)

Verschil operator ervoor (prefix) of operator erna (postfix):

```
int a, b;
a = 0; b = ++a;
printf ("%d %d\n", a, b);
a = 0; b = a++;
printf ("%d %d\n", a, b);
```

Output:

```
1 1
1 0
```

Dus:
++a retouneert waarde na increment en **a++** waarde voor increment

Assignment operators

De assignment = is ook een operator welke een waarde oplevert:

```
b = 2;
c = 3;
a = b + c;
```

is gelijk aan:

```
a = (b = 2) + (c = 3);
```

En

```
a = b = c = 0;
```

is gelijk aan:

```
a = (b = (c = 0));
```

Verder:

```
k += 2;
```

is gelijk aan:

```
k = k + 2;
```

Operator prioriteit en associativiteit

Prioriteit:

`1 + 2 * 3` Omdat * een hogere prioriteit heeft dan + moet dit gelezen worden als `1 + (2 * 3)`

Associativiteit:

`1 + 2 – 3` Omdat + en – gelijke prioriteit hebben, en omdat de associativiteit die bij + en – hoort “links naar rechts” is, moet dit gelezen worden als `((1 + 2)– 3)`

Overzicht prioriteit en associativiteit

hogere prioriteit

operator	associativiteit
() ++ (postfix) -- (postfix)	links naar rechts
+ (unary) - (unary) ++ (prefix) -- (prefix)	rechts naar links
* / %	links naar rechts
+ -	links naar rechts
= += -= *= /= etc.	rechts naar links

Dus:

```
z = ++a * b – c-- + d;
```

wordt geëvalueerd als:

```
z = ((((++a) * b) – (c--)) + d);
```

Voorbeeld: machten van 2

```
/* Some powers of 2 are printed. */

#include <stdio.h>

int main(void)
{
   int   i = 0, power = 1;

   while (++i <= 10)
      printf("%6d", power *= 2);
   printf("\n");
   return 0;
}
```

Output:

```
2     4     8    16    32    64   128   256   512  1024
```

Standard libraries

Gebruik van standard libraries

- De standard libraries bevatten vele nuttige functies en definities (zie ook Appendix A van het boek):
  - input/output
  - string manipulatie
  - maximum waarden voor int, float etc.
  - mathematische functies
  - geheugenbeheer
  - searching and sorting
  - etc.
- Om functies uit een standard library te gebruiken moet de desbetreffende header file geinclude worden:

  Bijv. `#include <math.h>`
- Hierin staan ook definities zoals INT_MAX (maximum waarde die past in een integer).
- De functie code wordt toegevoegd aan de executable tijdens het compileren

Voorbeeld gebruik standard library

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

int main(void)
{
   int   i, n;

   printf("How many random numbers ?: ");
   scanf("%d", &n);
   for (i = 0; i < n; ++i) {
      if (i % 10 == 0)
         putchar('\n');
      printf("%7d", rand());
   }
   printf("\n\n");
   return 0;
}
```

benodige header voor functie rand()

i % 10 geeft rest van i delen door 10, dus nieuwe regel na elke 10 waarden

elke nieuwe aanroep geeft een nieuw random getal

Input en output:

```
How many random numbers ?: 19

  16838   5758  10113  17515  31051   5627  23010   7419  16212   4086
   2749  12767   9084  12060  32225  17543  25089  21183  25137
```

Samenvatting

- Wat is programmeren ?
- Waarom C ?
- “Hello World!”: een simpel programma compileren en draaien
- Integrated Development Environment: Code::Blocks
- Kort overzicht van C
- Lexical elements/Syntax
- Operators
- Standard libraries

Agenda

- Vanmiddag: begin van practicum voor sommige groepen (gebouw Drebbelweg, zie informatie op Blackboard).
- Donderdagmiddag: begin practicum voor de andere groepen.
- Volgende week dinsdagochtend : 2e hoorcollege