Academie voor Technology, Innovation & Society Delft
Academie voor ICT & Media

Gestructureerd programmeren in C

GESPRG: File I/O, command line argumenten, typedef

DE HAAGSE HOGESCHOOL

Tekstbestanden

- Een tekstbestand bevat in ASCII-gecodeerde data (`char`'s).
- Tekstbestanden kunnen eenvoudig bewerkt worden (b.v. met Notepad. )
- Tekstbestanden kunnen ook met een C programma eenvoudig aangemaakt, beschreven en uitgelezen worden.

DE HAAGSE HOGESCHOOL

Tekstbestanden

- C kent een aantal functies met betrekking tot bestanden:

```
FILE *fopen(filename, mode);
```

- `fopen` geeft pointer naar een *file handle* terug, of `NULL` als de file niet benaderd kan worden. Voorbeeld:

```
FILE *fp = fopen("c:\\temp\\textfile.txt", "r");
```

In Windows: dubbele backslash

opent een bestand voor lezen, andere mogelijkheden voor `mode`: `"w"`, voor schrijven (huidige inhoud wordt gewist), `"a"` voor toevoegen (append).

DE HAAGSE HOGESCHOOL

Tekstbestanden

- Lezen uit een bestand:

```
fscanf(fp, "%d", &i);
```

- Schrijven naar een bestand:

```
fprintf(fp, "Variabele i is %d\n", i);
```

- Sluiten van een bestand:

```
fclose(fp);
```

DE HAAGSE HOGESCHOOL

Tekstbestand maken in C

```c
#include <stdio.h>

int main(void) {
    FILE* outfile;
    outfile = fopen("output.txt", "w");
    if (outfile == NULL) {
        printf("File output.txt kan niet aangemaakt worden.\n");
    }
    else {
        int i;
        for (i = 0; i < 10; i = i + 1) {
            fprintf(outfile, "Het kwadraat van %d is %d.\n", i, i * i);
        }
        fclose(outfile);
        printf("File output.txt is aangemaakt.\n");
    }
    getchar();
    return 0;
}
```

Een FILE* verwijst naar een bestand (file handle).

fopen opent een bestand in huidige map.

"w" (write) opent /maakt het bestand voor (over)schrijven.

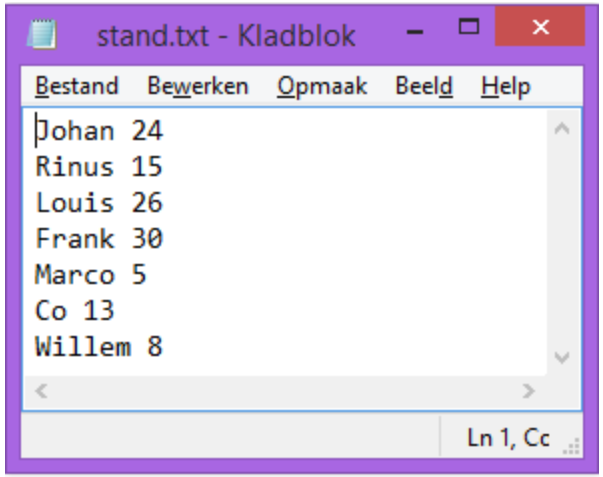
fopen geeft NULL terug als openen niet gelukt is. Daar **moet** op getest worden.

fprintf schrijft in een file.

fclose sluit een file.

Tekstbestand lezen in C

```c
#include <stdio.h>

int main(void) {
    char naam[80];
    int punten;
    FILE* infile;
    infile = fopen("stand.txt", "r");
    if (infile == NULL) {
        printf("File stand.txt kan niet gelezen worden.\n");
    }
    else {
        while (fscanf(infile, "%79s%d", naam, &punten) == 2) {
            printf("Speler %s heeft %d punten.\n", naam, punten);
        }
        fclose(infile);
    }
    getchar();
    return 0;
}
```

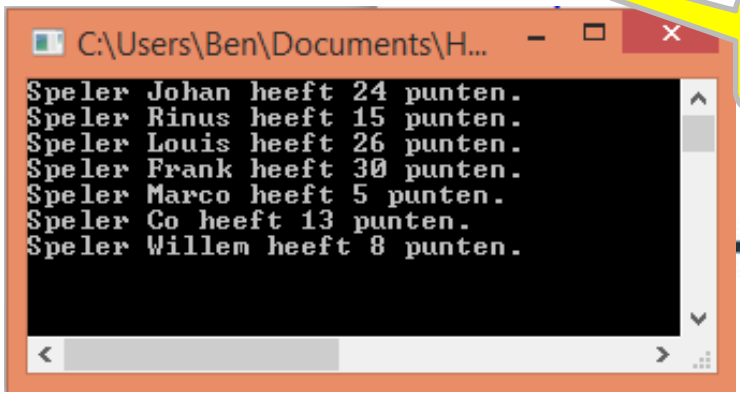


"r" opent de file voor lezen (read).

fscanf leest uit een file.

fscanf geeft aantal correct ingelezen variabelen terug.

%79s leest maximaal 79 karakters in.



HAAGSE HOGESCHOOL

Standaard geopende bestanden

- In een C-”omgeving” zijn altijd drie bestanden standaard geopend.

- `stdin`: de standaard invoer, het toetsenbord
- `stdout`: de standaard uitvoer, het beeldscherm
- `stderr`: de standaard error-uitvoer, het beeldscherm

- Je kan dus altijd lezen/schrijven van de standaard bestanden, tenzij je ze sluit met `fclose`.

```
fprintf(stderr, "Fout gedetecteerd!\n");
```

DE HAAGSE HOGESCHOOL

Command line argumenten

- Een C programma krijgt data mee van het OS.
- Deze data wordt in de volgende parameters gezet:

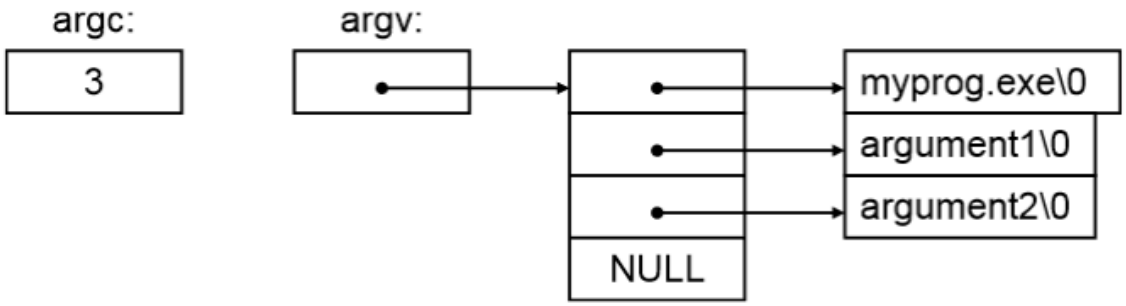
```
int main(int argc, char *argv[])
{
    /* rest of the code */
    return 0;
}
```

- De data wordt meegegeven bij het opstarten van het programma.

- `argc` is een integer.
- `argv` is een pointer naar een array van strings.

Command line argumenten

- Meegegeven data komt in `argc` en `argv`:



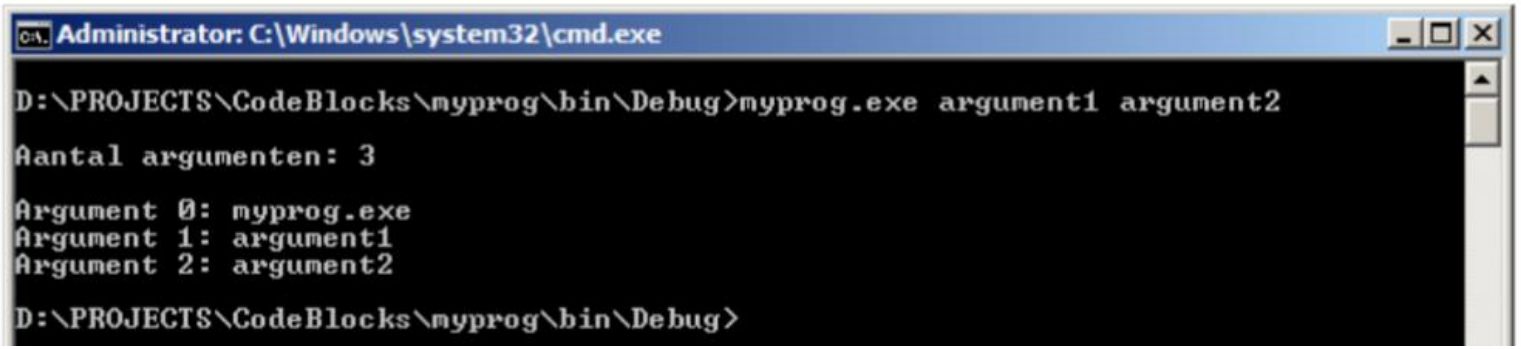
- Zie J.E.J. op den Brouw, *Command line argumenten en bestandsafhandeling in C* op Blackboard voor hoe je deze data kan gebruiken.

Command line argumenten

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

int main(int argc, char *argv[]) {

    int i;

    printf("\nAantal argumenten: %d\n\n", argc);
    for (i = 0; i<argc; i++) {
        printf("Argument %d: %s\n", i, argv[i]);
    }
    return 0;
}
```

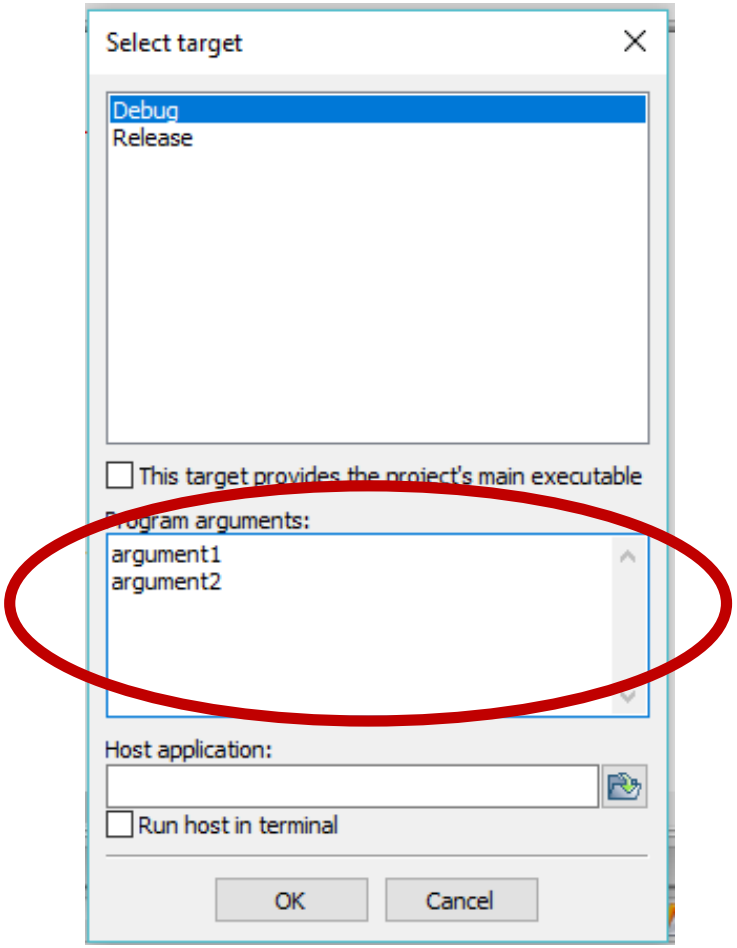
Command line argumenten

- Voorbeeld van het meegeven van data in een console:



Command line argumenten

- Voorbeeld van het meegeven van data in Code::Blocks:
  (selecteer eerst `Project -> Set programs' arguments`)



typedef

- De definitie van een variabele kan worden gescheiden van de declaratie:

```c
#include <stdio.h>

int main(void) {
   typedef int sok_t;  /* definitie */

   sok_t sok1 = 2, sok2 = 3;  /* declaratie */
   printf("sok1 = %d\nsok2 = %d", sok1, sok2);

   getchar();
   return 0;
}
```

typedef

- Typedefs worden vaak gebruikt bij structures:

```c
#include <stdio.h>
#include <stddef.h>

int main(void) {

   typedef struct Deelnemer {  /* definitie */
      char naam[80];
      int punten;
   } student_t;

   student_t s1 = {"Dennis", 7};  /* declaratie */
   printf("Student %s heeft %d punten\n", s1.naam, s1.punten);

   return 0;
}
```

DE HAAGSE HOGESCHOOL

C verkorte notatie

- Alle rekenoperatoren kunnen gecombineerd worden met operator= als het resultaat in de linker operand (rechts van het =-teken) moet worden opgeslagen.

```
a = a + b; ➔ a += b;
x = x * y; ➔ x *= y;
```

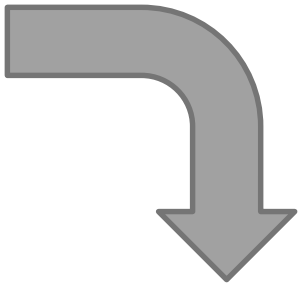
Deze assignment operatoren worden van rechts naar links geëvalueerd. Zie prioriteitentabel.

```
int a = 6, b = 7, c = 8;
a += b += c;
printf("a=%d b=%d c=%d\n", a, b, c);
(a += b) += c;
printf("a=%d b=%d c=%d\n", a, b, c);
```

DE HAAGSE HOGESCHOOL

C prioriteiten tabel (hoog naar laag)

Operator	Description	Associativity
( ) [ ] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	left-to-right
++ -- + - ! ~ (type) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of type) Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right



Operator	Description	Associativity
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

C verkorte notatie

- Increment en decrement operatoren.

```
a = a + 1; ➔ ++a; of a++;
b = b - 1; ➔ --b; of b--;
```

- Als een increment of decrement operator in een expressie wordt gecombineerd met meerdere operatoren dan wordt de prefix vooraf en de postfix achteraf uitgevoerd.

DE HAAGSE HOGESCHOOL

Postfix & Prefix

- Prefix wordt vooraf uitgevoerd en een postfix achteraf. Maar vooraf/achteraan aan wat?

```
#include <stdio.h>
int a = 6, b, c;

void main() {
  tmp();
  tmp();
}

void tmp(){
  b = a++;
  c = ++a;
  printf("a=%d b=%d c=%d\n",a,b,c);
  getchar();
}
```

- a=8 b=6 c=8
- a=10 b=8 c=10

C verkorte notatie

- Conditionele operator **?** **:**

```
int abs(int i) {
    if (i >= 0) {
        return i;
    }
    else {
        return -i;
    }
}
```

➔

```
int abs(int i) {
    return i >= 0 ? i: -i;
}
```

```
printf("Totaal: %d fout%s gevonden\n", errors,
        errors == 1 ? "" : "en");
```

DE HAAGSE HOGESCHOOL

Huiswerk

- Bestudeer J.E.J. op den Brouw, *Command line argumenten en bestandsafhandeling in C*
- Bestudeer boek/dictaat:
  - paragrafen 11.1 t/m 11.4
  - paragraaf 14.4
  - paragraaf 10.6
  - blz. 918
  - Paragraaf 3.11 en 3.12

Maak opdracht:

- 1 en 2 van https://www.w3resource.com/c-programming-exercises/file-handling/index.php
- 1 van https://www.indiabix.com/c-programming/typedef/

DE HAAGSE HOGESCHOOL

Academie voor Technology, Innovation & Society Delft
Academie voor ICT & Media

Gestructureerd programmeren in C

GESPRG: Preprocessor, compilatieproces

DE HAAGSE HOGESCHOOL

C preprocessor

- Wordt uitgevoerd voordat de “echte” compiler wordt gestart.
  - `#include` invoegen van andere bestanden
  - `#define` definiëren van macro’s.
  - `#if` conditionele compilatie

DE HAAGSE HOGESCHOOL

#include

- Include file bevat prototypes (code zit in library die meegelinkt wordt).

```
/*oops include vergeten!*/
int main(void) {
    printf("%.15lf", sin(1,2));
    getchar();
    return 0;
 }
```

Compiler:

Warning: 'printf' undefined
Warning: 'sin' undefined
Warning: 'getchar' undefined

Uitvoer:

```
0.000000000000000
```

DE HAAGSE HOGESCHOOL

#include

- Include file bevat declaraties (definities zitten in library die meegelinkt wordt).

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

int main(void) {
    printf("%.15lf", sin(1,2));
    getchar();
    return 0;
 }
```

Compiler:

Error: 'sin' : too many arguments for call

DE HAAGSE HOGESCHOOL

#include

- Include file bevat declaraties (definities zitten in library die meegelinkt wordt).

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

int main(void) {
    printf("%.15lf", sin(1.2));
    getchar();
    return 0;
 }
```

Uitvoer:

```
0.932039085967226
```

DE HAAGSE HOGESCHOOL

Zelfgemaakte include files

- `#include <stdio.h>`
  Zoek in de standaard include directories.

- `#include "homemade.h"`
  Zoek in de directory waar de .c file staat en daarna in standaard include directories.

- `#include "h:/mijn_includes/homemade.h"`
  Zoek in de directory h:/mijn_includes/.

- “Directory” = Unix/Linux-naam voor wat Windows “map” noemt.

DE HAAGSE HOGESCHOOL

Macro's (zonder argumenten)

- Zoek en vervang (voor het compileren).

```
#include <stdio.h>
#define AANTAL 10

int main(void) {
    int kwadraten[AANTAL];
    int i;
    for (i = 0; i < AANTAL; i = i + 1) {
        kwadraten[i] = i * i;
    }
}
```

Voordeel?

DE HAAGSE HOGESCHOOL

Macro's (met argumenten)

- Zoek en vervang (voor het compileren) met parameter(s).

```
#include <stdio.h>
#define KWAD(x) x * x

int main(void) {
    printf("%d\n", KWAD(8));
    printf("%d\n", KWAD(4 + 4));
    getchar();
    return 0;
 }
```

Uitvoer:

```
64
24
```

Wat is er mis?

DE HAAGSE HOGESCHOOL

Macro's (met argumenten)

- Zoek en vervang (voor het compileren) met parameter(s).

```
#include <stdio.h>
#define KWAD(x) ((x) * (x))

int main(void) {
    int i = 7;
    printf("%d\n", KWAD(8));
    printf("%d\n", KWAD(4 + 4));
    printf("%d\n", KWAD(++i));
    getchar();
    return 0;
}
```

Uitvoer:

```
64
64
81
```

Wat is er mis?

DE HAAGSE HOGESCHOOL

No Macro's (met argumenten)

- Beter: gebruik functies

```
#include <stdio.h>
int kwad(int x) {
       return x * x;
}
int main(void) {
    int i = 7;
    printf("%d\n", kwad(8));
    printf("%d\n", kwad(4 + 4));
    printf("%d\n", kwad(++i));
    getchar();
    return 0;
 }
```

Uitvoer:

```
64
64
64
```

DE HAAGSE HOGESCHOOL

Conditionele compilatie

- `#if`
- `#ifdef`
- `#ifndef`
- `#elif`
- `#else`

- `#endif`

DE HAAGSE HOGESCHOOL

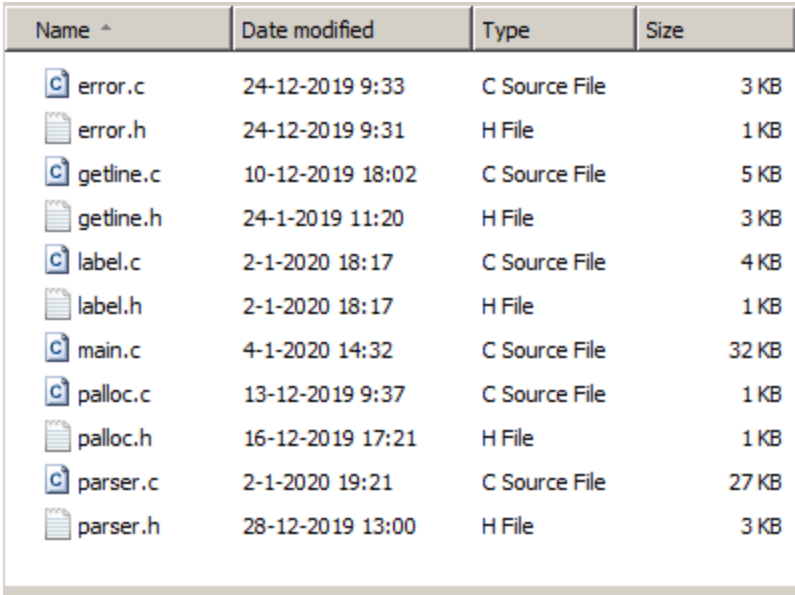
Conditionele compilatie

```
#include <stdio.h>

#define DEBUG 1

int main(void) {
…
#if DEBUG
        printf("DEBUG: line = %s\n", str);
#endif
…
```

DE HAAGSE HOGESCHOOL

Compilatieproces

- In een groot project wordt code meestal over meerdere .c- en .h-bestanden verspreid.



Name	Date modified	Type	Size
error.c	24-12-2019 9:33	C Source File	3 KB
error.h	24-12-2019 9:31	H File	1 KB
getline.c	10-12-2019 18:02	C Source File	5 KB
getline.h	24-1-2019 11:20	H File	3 KB
label.c	2-1-2020 18:17	C Source File	4 KB
label.h	2-1-2020 18:17	H File	1 KB
main.c	4-1-2020 14:32	C Source File	32 KB
palloc.c	13-12-2019 9:37	C Source File	1 KB
palloc.h	16-12-2019 17:21	H File	1 KB
parser.c	2-1-2020 19:21	C Source File	27 KB
parser.h	28-12-2019 13:00	H File	3 KB

- In de .h-bestanden staan definities van structures/array's en prototypes van functies.
- De "echte" code staat in de .c-bestanden
- Een groot aantal bekende functies staat in zogenoemde *bibliotheken* (zoals `strlen`, `printf`, `sin`). Die zijn al gemaakt.

DE HAAGSE HOGESCHOOL

Compilatieproces

- Alle .c-bestanden worden eerst omgezet in zogenoemde objectbestanden met de extensie .o:

```
gcc -c parser.c –o parser.o
```

Object-bestand, kan niet gestart worden.

- Daarna worden alle .o-bestanden én bibliotheken met elkaar *gelinkt*.

```
gcc –o mic32asm.exe main.o parser.o label.o libc.a
```

De GNU C-compiler

Uitvoerbaar bestand

De standaard C-bibliotheek

- Ontwikkelomgevingen regelen het compilatieproces automatisch.

DE HAAGSE HOGESCHOOL

Huiswerk

- Bestudeer boek/dictaat:
  - paragrafen 13.1 t/m 13.3, 13.4, 13.4.1, 13.4.2, 13.5

DE HAAGSE HOGESCHOOL