4EE11 – Project
Programmeren voor W

College 2, 2008–2009, Blok D

Tom Verhoeff, Software Engineering & Technology, TU/e

Onderwerpen

- Terugblik
- Functies
- Organisatie (architectuur) van programma's
- Arrays
- Structuren
- File I/O
- IDE handigheden

Onbedoelde fout

- In `test_grades.c`, regel 23,

```
of = fopen("test.log","r");
```

- moet zijn

```
of = fopen("test.log","w");
```

Fouten zoeken

- Concentreer je op de eerste foutmelding
- De rest kan gevolgschade zijn en vanzelf verdwijnen als je de eerste fout oplost
- Niet te lang modderen; vraag assistentie

Functies

- Bundelt stel opdrachten
- Aanroepen (gebruiken; meermalen) versus implementeren (definiëren; eenmalig)
- Parameters: namen, types, smaken, volgorde
- Resultaat: type, of niet van toepassing (void)
- Lokale versus globale variabelen
- Doel (specificatie, contract)
- Abstraheert van implementatiedetails

Functie: C voorbeeld

```
#include <stdio.h>

void print_tafel ( int n )
  // pre:  0 <= n <= 99
  // post: tafel van n is afgedrukt op stdout
{
  for ( int regel = 1; regel <= 10; ++regel ) {
    int uitkomst = regel * n; // uitvoer
    printf ( "%2d x %2d = %4d\n", regel, n, uitkomst );
  }
}

int main ( void )
{
  int grondtal; // tafelgrondtal (invoer)

  printf ( "Geef grondtal voor tafel: " );
  scanf("%d", &grondtal);

  print_tafel ( grondtal );

  return 0;
}
```

Functie: C voorbeeld

```
#include <stdio.h>

void print_tafel ( int n )
  // pre:  0 <= n <= 99
  // post: tafel van n is afgedrukt op stdout
{
  for ( int regel = 1; regel <= 10; ++regel ) {
    int uitkomst = regel * n; // uitvoer
    printf ( "%2d x %2d = %4d\n", regel, n, uitkomst );
  }
}

int main ( void )
{
  int grondtal; // tafelgrondtal (invoer)

  printf ( "Geef grondtal voor tafel: " );
  scanf("%d", &grondtal);

  print_tafel ( grondtal );

  return 0;
}
```

Interface = prototype + contract

Implementatie

Aanroep

Functie als functie

- `int sqr ( int a ) { return a*a; }`
- functie: *`type name ( parameters )`*
- gebruiken als formule met type *`type`*:
  … `name ( argumenten )` …
- `sqr(1) + sqr(2) + sqr(3)`

Functie als procedure

- procedure: **`void`** *`name`* `(` *`parameters`* `)`
- procedure gebruiken als opdracht:
  `name` `(` *`argumenten`* `);`

- Zonder parameters: **`void`** *`name`* `(` **`void`** `)`
- Dan aanroepen als: *`name`* `( );`

Functie documenteren

- preconditie: voorwaarde die moet gelden bij aanroep; plicht van aanroeper
- resultaat (van functie): uitgedrukt in termen van parameters
- postconditie (van procedure): uitspraak die zal gelden na aanroep

Tweezijdig contract

partij \ contract	Preconditie	Resultaat, Postconditie
Aanroeper	Plicht ↓	Nut ↑
Implementator	Nut →	Plicht

Aanroeper wil abstraheren van →

Verdelen over files

```
/* ========== file mylib.h ==========*/
void print_tafel ( int n ); // function prototype
  // pre:  0 <= n <= 99
  // post: tafel van n is afgedrukt op stdout
```

Interface tussen twee werelden

```
/* ========== file mylib.c ==========*/
#include <stdio.h>

void print_tafel ( int n ) {
  for ( int regel = 1; regel <= 10; ++regel ) {
    int uitkomst = regel * n; // uitvoer
    printf ( "%2d x %2d = %4d\n", regel, n, uitkomst );
  }
}
```

Wereld van de implementator

```
/* ========== file tafels.c ==========*/
#include <stdio.h>
#include “mylib.h”

int main ( void ) {
  int grondtal; // tafelgrondtal (invoer)

  printf ( "Geef grondtal voor tafel: " );
  scanf("%d", &grondtal);
  print_tafel ( grondtal );
  return 0;
}
```

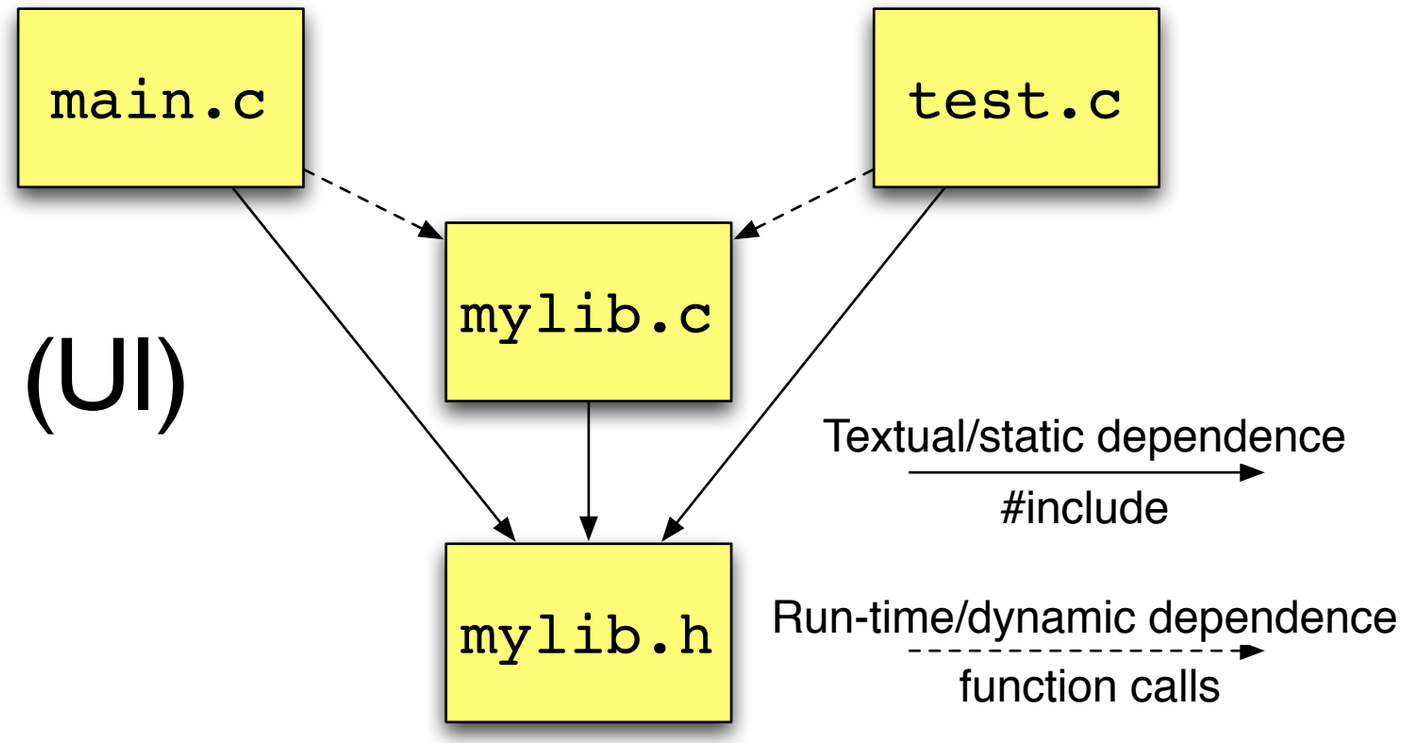
Wereld van de aanroeper

Testen

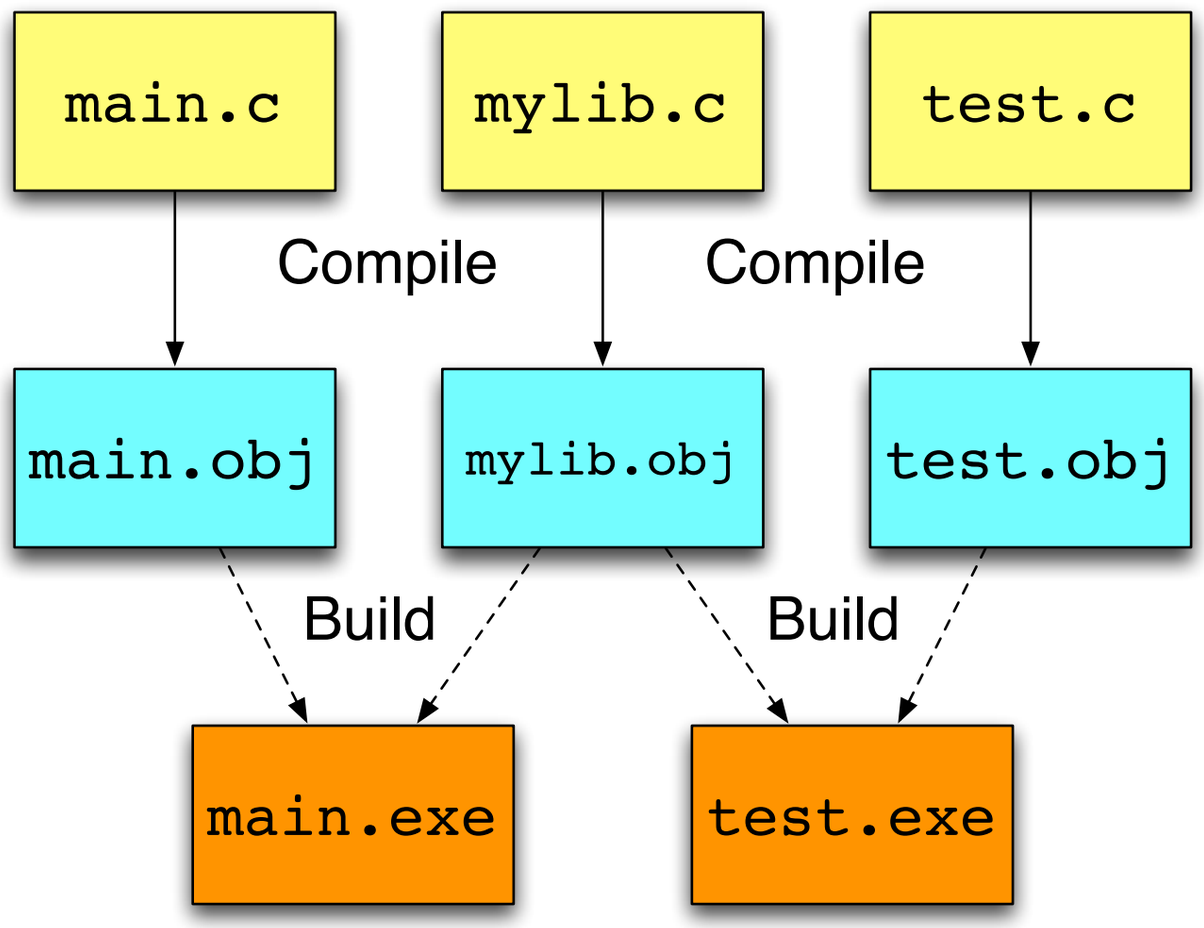
```
/* ========== file test_tafels.c ==========*/

#include <stdio.h>
#include “mylib.h”

int main ( void )
{
  print_tafel ( 1 );
  print_tafel ( 10 );

  return 0;
}
```

Pelles: files in projecten

- Header files
- Library files
- Main program (UI)
- Test driver(s)
- Executables



Compile & Build



Array

- Bundelt variabelen van *hetzelfde* type
- `int A[10];`
- `A[0]` t/m `A[9]` zijn van type `int`
- Array indexering in C altijd vanaf 0
- Index-out-of-bounds: A[i] met i < 0 || i > 9
- I.o.o.b. geeft vaak geen run-time foutmelding, maar is wel (ernstige) fout; oorzaak van veel beveiligingslekken

Array doorlopen

```
// polynoomevaluatie (Horners schema)
int c[10] = { … }; // coefficienten
double x; // argument
double y = 0.0; // waarde
scanf ( “%f”, &x );

for ( int i = 9; i >= 0; --i ) {
  y = c[i] + x * y;
}

// y == som i in 0 t/m 9: c[i] * x^i
printf ( “%1.5f\n”, y );
```

Pointers, adressen

```
int v = 42; // declaratie en initialisatie
int a[3] = { 7, 666 };
int *p; p = &v; // &v is adres van v

// type van p is pointer naar int
// waarde van p is adres van v

// type van *p is int
// waarde van *p is waarde op adres p,
//   waarde waar p naar wijst
// *p is hier dezelfde variabele als v
// na *p = 17; geldt v == 17
// aliasing: twee verschillende namen
//   voor hetzelfde ding
// I.h.a. geldt  *&v == v  en  &*p == p

// p+1 is eerstvolgende adres na p
// p+i is i adressen na p
```

Geheugen		
Adres	Waarde	Var
1000	42	v
1001	7	a[0]
1002	666	a[1]
1003	*undef*	a[2]
1004	1000	p

Array is a pointer

- int *p; // p is *pointer naar* int; *p is een int
- int A[10]; // A is pointer naar rij van 10 int's
- A is contante; A == &A[0], A[i] == *(A + i)
- *p == p[0]
- Zoek 0 in A, wetende dat die vóórkomt:
  - for (int i = 0; A[i] != 0; ++i) ;
  - for (p = A; *p != 0; ++p) ;

Reference parameters

```
void verdubbel ( int a, int *r )
  // pre:  none
  // post: *r == a + a
{
  *r = a + a; // schrijft a+a op adres r
}

...

int b; // om resultaat van verdubbel op te vangen

verdubbel ( 1, &b ); // geef adres voor resultaat mee
// b == 2
```

Tafel in array

```
#include <stdio.h>

void generate_tafel ( int n, int *t )
  // pre:  0 <= n <= 99, t wijst naar integer array met 11 plaatsen
  // post: t bevat tafel van n: t[0] == n, t[i] == i * n (1 <= i <= 10)
{
  for ( int regel = 1; regel <= 10; ++regel ) {
    t [ regel ] = regel * n;
  }
}

void print_tafel ( const int *t )
  // pre:  t wijst naar array met tafel,
  //       d.w.z. t[i] == i * t[0] voor 1 <= i <= 10
  // post: tafel t is afgedrukt op stdout
{
  for ( int regel = 1; regel <= 10; ++regel ) {
    printf ( "%2d x %2d = %4d\n", regel, t[0], t[regel] );
  }
}
```

Type een naam geven

- **`typedef`** …*`NAAM`*…**`;`**
- waarbij …`v`…`;` een goedgetypeerde declaratie van `v` moet zijn
- *`NAAM`* `v;` equivalent met …`v`…`;`
- **`typedef int`** `TAFEL[11];`
- `TAFEL t;` equivalent met **`int`** `t[11];`

Tafel als type

```
#include <stdio.h>

typedef int TAFEL[11]; // tafel van t[0]
  // invariant voor TAFEL t: t[i] == i * t[0] voor 1 <= i <= 10

void generate_tafel ( int n, TAFEL t ) {
  // pre:  0 <= n <= 99
  // post: t bevat tafel van n

  t[0] = n;

  for ( int regel = 1; regel <= 10; ++regel ) {
    t [ regel ] = regel * n;
  }

}

void print_tafel ( const TAFEL t ) {
  // pre:  none
  // post: tafel t is afgedrukt op stdout

  for ( int regel = 1; regel <= 10; ++regel ) {
    printf ( "%2d x %2d = %4d\n", regel, t[0], t[regel] );
  }

}
```

Meerdimensionaal array

- `int matrix[10][10];`
- `matrix[i]` is een hele rij
- `matrix[i][j]` is een los element

Strings

- Een *string* is een rijtje karakters van type **`char`** afgesloten met het *nulkarakter* `'\0'`
- Een string wordt opgeslagen in een array of aangewezen door een pointer
- `char *s = "Tom";`
- `s[0] == 'T', s[4] == '\0'`

Struct in-/uitpakken

- Bundelt variabelen, mogelijk van *verschillend* type
- `struct { char lijn; int rij; } v;`
  - variabele `v.lijn` heeft type **`char`**
  - variabele `v.rij` heeft type **`int`**
  - `typedef struct { char lijn; int rij; } SCHAAKVELD;`
  - `SCHAAKVELD v;`
  - `SCHAAKVELD verplaatsing[2];`

Complexere types

- `struct { float x, y; } A[10];`
- `A` is array van 10 structs van floats x en y
- `A[i]` is een struct met floats x en y
- `A[i].x` is een float
- `struct { float x[10], y[10]; } B;`
- `B` is struct met arrays x en y van 10 floats
- `B.x` is een array van 10 floats
- `B.x[i]` is een float

Globale variabelen

```
typedef struct { float x, y; } MEGAGON[1000000];
MEGAGON my_global_megagon;

float f ( MEGAGON p ) {
  float h;
  ... p ... my_global_megagon ...
}

void g ( MEGAGON p ) {
  float r = f ( p ); ...
}

int main ( void ) {
  float a;
  g ( my_global_megagon );
}
```

- M.n. voor grotere datastructuren
- Adressen ervan doorgeven i.p.v. kopiëren

Geheugenindeling

Heap	…
free vvvv	…
free ^^^^	…
Stack frame f	p, h
Stack frame g	p, r
Stack frame main	a
Globalen	my_global_megagon

Stack frame bestaat alleen als functie actief is.

Stack frame bevat ook het "*terugkeeradres*".

File I/O

- Zie algemene opdracht, `test_grades.c`
- File descriptor declareren, van type ‘pointer naar FILE’: `FILE *myfile;`
- File openen voor lezen (met `"r"`) of schrijven (met `"w"`)
- fopen, fscanf, fprintf, fflush, fclose
- stdin, stdout, stderr

Pelles IDE

- Monospaced font (`Courier`) instellen
- Set active project in Project menu, of …
- Foutmelding localiseren via dubbelklikken
- Help-informatie opvragen via rechtermuisknop (op woord in editor)
- Source line numbers laten zien
- Debugger: breakpoint, waarde inspecteren

Volgende week

- Programma opzet en ontwerp
- Systematisch testen