Academie voor Technology, Innovation & Society Delft

Academie voor ICT & Media

Gestructureerd programmeren in C

GESPRG: Functiepointers en details

DE HAAGSE HOGESCHOOL

Pointers naar functies

- In C kun je een pointer naar een functie definiëren.
  - De waarde van de pointer is het beginadres (van de code) van de functie.

```c
#include <stdio.h>

int kwadraat(int c) {
    return c * c;
}

int dubbel(int c) {
    return c + c;
}

int main(void) {
    int a = 7, b;
    int (*pnf)(int);
    pnf = &dubbel;
    b = (*pnf)(a);
```

pnf is een **pointer naar een functie** met een int als parameter en een int returnwaarde

pnf **wijst naar** de functie dubbel (pnf wordt gelijk aan **het adres van** de functie dubbel)

Waarom haakjes?

DE HAAGSE HOGESCHOOL

Pointers naar functies

- Verkorte schrijfwijze.
  - Naam van een functie ⬅➡ beginadres (van de code) van de functie.

```c
#include <stdio.h>

int kwadraat(int c) {
    return c * c;
}

int dubbel(int c) {
    return c + c;
}
```

```c
int main(void) {
    int a = 7, b;
    int (*pnf)(int);
    pnf = dubbel;
    b = pnf(a);
```

DE HAAGSE HOGESCHOOL

Pointers naar functies

- Wat is het nut?
  - Functie als parameter.

```
#include <stdio.h>
/* … */
void printTabel(int (*p)(int), int van, int tot, int stap) {
    int x;
    for (x = van; x < tot; x += stap) {
        printf("%10d %10d\n", x, (*p)(x));
    }
}
int main(void) {
    printf("De kwadraten van 1 t/m 10\n");
    printTabel(&kwadraat, 1, 11, 1);
    printf("De dubbelen van de drievouden van 0 t/m 30\n");
    printTabel(&dubbel, 0, 31, 3);
```

DE HAAGSE HOGESCHOOL

Uitvoer

```
De kwadraten van 1 t/m 10
         1          1
         2          4
         3          9
         4         16
         5         25
         6         36
         7         49
         8         64
         9         81
        10        100
De dubbelen van de drievouden van 0 t/m 30
         0          0
         3          6
         6         12
         9         18
        12         24
        15         30
        18         36
        21         42
        24         48
        27         54
        30         60
```

Details!

- The devil is in the details.

DE HAAGSE HOGESCHOOL

Type Specifiers (C89)

- `short int` — `sizeof(short)` ≤ `sizeof(int)` meestal 2
- `long int` — `sizeof(long)` ≥ `sizeof(int)` meestal 4
- `signed int` / `signed char` — two's complement values
- `unsigned int` / `unsigned char` — only values ≥ 0

Verwarrend!

DE HAAGSE HOGESCHOOL

(Extra) Type Specifiers (C99)

- `stdint.h`
  - `int8_t` en `uint8_t`
  - `int16_t` en `uint16_t`
  - `int32_t` en `uint32_t`
  - `int64_t` en `uint64_t`

Worden zeer vaak gebruikt bij programma's waar exacte grootte van variabelen gewenst is, bijv. bij microcontrollers.

- `stdbool.h`
  - `bool` en de constanten: `false` en `true`
- `complex.h`
  - `complex` en diverse functies

DE HAAGSE HOGESCHOOL

Block Scope (C89)

- Variabelen mogen alleen aan het begin van een compound statement gedefinieerd worden, dus na `{`.
- De scope (zichtbaarheid) is het betreffende compound statement behalve als naam verborgen is (door variabele met dezelfde naam).
- De lifetime (levensduur) tot einde van compound statement `}` uitgevoerd is.

DE HAAGSE HOGESCHOOL

Block Scope (C99)

- Variabelen mogen overal in een compound statement gedefinieerd worden.
- De scope (zichtbaarheid) is tot einde van het betreffende compound statement behalve als naam verborgen is (door variabele met dezelfde naam).
- De lifetime (levensduur) tot einde van compound statement } uitgevoerd is.

```
int a[] = {1, 2, 3, 4, 5} , som = 0;
for (int i = 0; i < sizeof a / sizeof a[0]; i++) {
        som += a[i];
}
```

DE HAAGSE HOGESCHOOL

break

- break
  - Verlaten switch
  - Verlaten for, while of do … while
  - Niet voor de if!

Komt de duidelijkheid meestal niet ten goede!

DE HAAGSE HOGESCHOOL

continue

- `continue`
  - Ga meteen naar test `for`, `while` of `do` … `while`

Komt de duidelijkheid meestal niet ten goede!

DE HAAGSE HOGESCHOOL

Tel letters eerste woord

```c
#include <stdio.h>

int main(void) {
    char zin[] = "Hallo daar";
    int i;
    for (i = 0; zin[i] != ' '; i++) /* nothing to do */ ;
    printf("Lengte eerste woord = %d\n", i);
    getchar();
    return 0;
}
```

Wie ziet het probleem ?

Soms heb je aan 1 woord genoeg

DE HAAGSE HOGESCHOOL

Tel letters eerste woord

```c
#include <stdio.h>

int main(void) {
    char zin[] = "Hallo daar";
    int i;
    for (i = 0; zin[i] != ' '; i++) {
        if (zin[i] == '\0')
            break;
    }
    printf("Lengte eerste woord = %d\n", i);
    getchar();
    return 0;
}
```

Werkt wel maar is niet zo duidelijk!

DE HAAGSE HOGESCHOOL

Tel letters eerste woord

```c
#include <stdio.h>

int main(void) {
    char zin[] = "Hallo daar";
    int i;
    for (i = 0; zin[i] != ' ' && zin[i] != '\0'; i++) {
        /* nothing to do */
    }
    printf("Lengte eerste woord = %d\n", i);
    getchar();
    return 0;
}
```

Werkt ook!

DE HAAGSE HOGESCHOOL

Huiswerk

- Paragraaf 7.12, 4.9, 5.11
- Bekijk (eventueel):
  - http://en.wikibooks.org/wiki/C_Programming/C_Reference/stdint.h
  - http://en.wikibooks.org/wiki/C_Programming/C_Reference/stdbool.h
  - http://en.wikibooks.org/wiki/C_Programming/C_Reference/complex.h
- Maak opdracht:
  - 22 en 33 van https://www.w3resource.com/c-programming-exercises/for-loop/index.php
  - Program 1 en program 2 van https://www.geeksforgeeks.org/enumeration-enum-c/

DE HAAGSE HOGESCHOOL