Nybegynnerkurs i C

Øyvind Grønnesby

14. oktober 2004

Introduksjon

- pass-by-value
- svakt typet
- “portabel assembler”
- siste ISO-standard er C99

Hello world

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

/* et komplett program!
 * dette trenger et prosjekt åp sourceforge.
 */

int main(void)
{
    printf("Hello, world.\n");
    exit(EXIT_SUCCESS);
}
```

Argumenter

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

int main(int argc, char *argv[])
{
    for (int i = 0; i < argc; ++i) {
        printf("Hello, %s!\n", argv[i]);
    }
    exit(EXIT_SUCCESS);
}
```

Kompilering: UNIX

```
$ gcc -o hello-integer hello-integer.c
$ ./hello-integer
Hello integer! (10)

$ gcc -Wall -std=c99 -pedantic \
> -o hello-integer hello-integer.c
$ ./hello-integer
Hello integer! (10)
```

Deklarering

Skriver ut: `Hello integer! (10)`

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

int main(void)
{
    unsigned int i = 10;

    printf("Hello integer! (%d)\n", i);
    exit(EXIT_SUCCESS);
}
```

Heltall

- `short`, `int`, `long`, `long long`
- Kan kombineres med `unsigned`
- Varierer i størrelse fra 16 til 64 bits
- `0x08`, `8`, `010`, `8L`, `8UL`

Flyttall

- `float`, `double`, `long double`
- Informasjon om grenseverdier for tallene i `<float.h>`
- Skal i C99 bruke IEEE 754
- `8.0`, `8e0`, `8.0L`

Tekst

- `char` er typisk på 8 bit og inneholder ett tegn
- `wchar_t` er mer oppdatert for locale
- Tekst er alltid arrays med `char` eller `wchar_t`
- `'a', "Hei og hopp!"`

Tekst - et eksempel

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

int main(void)
{
    char c;
    while ((c = getchar()) != NULL)
        putchar(c);
    exit(EXIT_SUCCESS);
}
```

struct

```
struct node {
    int value;
    struct node *next;
};

void node_ab(void) {
    struct node a, b;

    a.value = 10, b.value = 10;
}
```

Operatorer

- Tilordning: `=`, `+=`, `*=`, `/=`, `|=` , `&=`, `^=`
- Aritmetikk: `+`, `-`, `*`, `/`, `++`, `-`
- Logikk: `&&`, `||`
- Binær: `|`, `&`, `^`, `«`, `»`

if

```
if (elephant_is_nice(dumbo) == 0) {
    picture = take_picture(dumbo);
}

if (elephant_is_nice(dumbo) == 0) {
    picture = take_picture(dumbo);
}
else {
    dumbo = apply_makeup(dumbo);
}
```

while

```
while (elephant_is_nice(dumbo) != 0) {
    dumbo = apply_makeup(dumbo);
}
```

do ... while

```
do {
    dumbo = apply_makeup(dumbo);
} while (elephant_is_nice(dumbo) != 0);
```

for

- Er på formen `for (init; test; slutt) { kode }`
- Ekvivalent med: `kode; while (test) { kode; slutt; }`

```
for (int i = 0; i < 10; i++) {
    printf("Jeg har %d elefanter!\n", i);
}
```

switch

```
switch (character) {
    case 'a':
        i++; break;
    case 'b':
    case 'c':
        i++; break;
    case 'd'
        i++;
    case 'e':
        i++; break;
    default:
        --i;
}
```

Pekere

- En peker er en adresse til et minneområde
- Deklareres med `int *a`
- Pekeren til en gitt variabel fås med `&a`
- Objektet til pekeren er `*a`

Pekere, et eksempel

Ta imot en peker til en integer og inkrementer tallet med en.

```
void increment(int *a)
{
    *a++;
}

int main(void)
{
    int i = 10;
    increment(&i);
    exit(EXIT_SUCCESS);
}
```

malloc

- Allokerer minne, hvis det er ledig
- Veldig sprø. Sjekk alltid returverdi
- Søker lineært etter første ledige minneblokk som er stor nok

malloc - eksempel

```
node *append_node(int value, node *last)
{
    node *new;
    new = malloc(sizeof(*new));
    if (new == NULL) {
        return NULL;
    }
    else {
        new->value = value;
        new->next  = NULL;
        last->next = new;
        return new;
    }
}
```

free

- `free()` frigjør preallokert minne ved gitt peker
- Kan skape problemer hvis minnet ved adressen er allerede frigjort
- Trenger like mye pass som `malloc()`

free - eksempel

```
void free_list(node *head)
{
    node *current;
    while (node != NULL) {
        current = node;
        node    = node->next;
        free(current);
    }
}
```

Array

- Deklareres med `type name[size]`
- Referer til element `i` med `name[i]`
- `name` er en peker til første element
- `(name+i)` er en peker til element `i`, som betyr at `*(name+i)` er ekvivalent med `name[i]`

Array - eksempel

```
#define MAX_SIZE (1<<10)

static char a[MAX_SIZE];

void read_array(void)
{
    for (int i = 0; i < MAX_SIZE; i++) {
        a[i] = getchar();
    }
}
```

Dynamiske array

- Siden pekere og array er så nært knyttet til hverandre kan de allokeres enkelt til vilkårlig størrelse.

```
int *make_int_array(int size)
{
    int *a;
    a = malloc(sizeof(*a) * size);
    if (a != NULL) {
        memset(a, 0, sizeof(*a));
        return a;
    } else {
        return NULL;
    }
}
```

Makroer

- Gir kontroll over kodegenerering før koden sende til kompilator

#include

- Inkluderer en fil inn i kildekoden, ofte brukt for header-filer
- Med syntaks `<fil>` vil den se etter filen på et kompilatorspesifikt sted
- Med fnutter - "`fil`" - er det relativ eller absolutt sti

#include - eksempel

```
#include <stdlib.h>
#include <linux/limits.h>
#include "myheader.h"
#include "../otherlib/header.h"
```

#ifdef, #ifndef

- Tester om en makro allerede er definert

```
#ifndef MYHEADER_H
#define MYHEADER_H

int myfunction(int a, int b);

#endif /* MYHEADER_H */
```

#define

- Definerer en ren substitusjon av verdier. Ofte brukt for å deklarere globale verdier

```
#define MAX_ARRAY_SIZE (1<<10)
```

```
#define increment_node(x) ((x)->value++)
```

Lenker

- `http://www.eskimo.com/~scs/C-faq/top.html`
- The C Programming Language, Kernighan & Ritchie

Duff's Device

```
send(to, from, count)
register short *to, *from;
register count;
{
        register n=(count+7)/8;
        switch(count%8){
        case 0: do{ *to = *from++;
        case 7:         *to = *from++;
        case 6:         *to = *from++;
        case 5:         *to = *from++;
        case 4:         *to = *from++;
        case 3:         *to = *from++;
        case 2:         *to = *from++;
        case 1:         *to = *from++;
                }while(--n>0);
}
```