

# Sommerkursus: Introduktion til C

Peter Severin Rasmussen

SDU

13. august 2016

# Plan for i dag

## Del 1 (klokken 10.15-12.00)

- ▶ Oversættelse og afvikling af et program
- ▶ Variable, datatyper og aritmetiske udtryk
- ▶ Forgreninger
- ▶ Løkker
- ▶ Arrays

## Del 2 (klokken 13.15-16.00)

- ▶ Funktioner
- ▶ Strukturer
- ▶ Pointere (“pegepinde”)
- ▶ Input og output

## Kodecafe (klokken 16.15-18.00)

Del 1

## Oversættelse og afvikling af et program

```
#include <stdio.h>

int main() {
    printf("Hello world!\n");
    return 0;
}
```

```
$ gcc program.c
```

```
$ ./a.out
```

```
# ↪ Hello world!
```

## Oversættelse og afvikling af et program

```
#include <stdio.h>

int main() {
    printf("Hello world!\n");
    return 0;
}
```

```
$ gcc program.c -o program
```

```
$ ./program
```

```
# ↪ Hello world!
```

## Oversættelse og afvikling af et program

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

int main(int argc, char** argv) {
    ... // <- Focus
    return 0;
}
```

## Forskelle på C og Java

```
public class Hello {  
    public static void main(String args[]) {  
        System.out.println("Hello world!");  
    }  
}
```

```
#include <stdio.h>  
int main(int argc, char** argv) {  
    printf("Hello world!\n");  
    return 0;  
}
```

## Forskelle på C og Java

```
public class FizzBuzz {  
    public static void main(String args[]) {  
        int lower = 1;  
        int upper = 100;  
        for (int i = lower; i < upper; i++) {  
            if (i % 3 == 0 || i % 5 == 0) {  
                System.out.println(" " + i);  
            }  
        }  
    }  
}
```

```
#include <stdio.h>  
int main(int argc, char** argv) {  
    int lower = 1;  
    int upper = 100;  
    for (int i = lower; i < upper; i++) {  
        if (i % 3 == 0 || i % 5 == 0) {  
            printf(" %d\n", i);  
        }  
    }  
    return 0;  
}
```



## Variable, datatyper og aritmetiske udtryk

```
int n;  
n = 5;
```

## Variable, datatyper og aritmetiske udtryk

```
int n;  
n = 5;
```

```
int n = 5;
```

## Variable, datatyper og aritmetiske udtryk

```
int n;  
n = 5;
```

```
int n = 5;
```

Værdien af `n` inden `n = 5`?

# Variable, datatyper og aritmetiske udtryk

## Typer

```
int, long,  
char,  
bool,  
float, double,  
size_t, ...
```

## Variable, datatyper og aritmetiske udtryk

```
int a = 18;
```

```
int b = 24;
```

```
int c = a + b;
```

## Variable, datatyper og aritmetiske udtryk

```
int a = 18;
```

```
int b = 24;
```

```
int c = a + b;
```

```
bool p = true;
```

```
bool q = false;
```

```
bool r = p && q;
```

## Variable, datatyper og aritmetiske udtryk

```
int a = 18;  
int b = 24;  
int c = a + b;
```

```
bool p = true;  
bool q = false;  
bool r = p && q;
```

+, -, \*, /, %

&&, ||

==, !=

# Forgreninger

```
int a = 13;  
if (a > 10) {  
    printf("Success!\n");  
}
```



## Forgreninger

```
int a = 13;
bool b = true;
if (a < 20 && !b) {
    printf("Success!\n");
} else {
    printf("Fail!\n");
}
```

## Forgreninger

```
int n = 99;
switch(n) {
    case 97:
    case 98:
        printf("Almost!\n");
        break;
    case 99:
        printf("We are there!\n");
        break;
    case 100:
        printf("We have gone to far!\n");
        break;
    default:
        printf("Not of interest!\n");
        break;
}
```

## Løkker

```
int a = 5;
int l = 0;
while (a > 1) {
    printf("%d\n", a);
    a = a / 2;
    l++;
}
printf("%d\n", l);
```

## Løkker

```
int a = -5;
do {
    printf("%d\n", a);
    a = a / 2;
} while (a > 1);
printf("%d\n", a);
```

## Løkker

```
for (int i = 0; i < 20; i++) {  
    printf("%d\n", i);  
}
```

# Arrays

```
int arr[10];
```

# Arrays

```
int arr[10];  
arr[0] = 17;  
arr[1] = 25;  
arr[9] = arr[0] + arr[1];
```

# Arrays

```
int arr[10];  
arr[0] = 17;  
arr[1] = 25;  
arr[9] = arr[0] + arr[1];  
int b[] = {1, 2, 3};
```



# Arrays

## Strings

```
char c[] = {'a', 'b', 'c', '\0'};
```

# Arrays

## Strings

```
char c[] = "abc";
```

## printf

```
printf("Her er du:\n");  
printf("By:      %s\n",    "Odense");  
printf("Post nr.: %d\n",    5000);  
printf("Grader:   %f °C\n", 17.3);
```

## printf

```
printf("Her er du:\n");  
printf("By:      %s\n",    "Odense");  
printf("Post nr.: %d\n",    5000);  
printf("Grader:   %f °C\n", 17.3);
```

```
printf("Grader:   %.1f °C\n", 17.3);  
printf("Procent:  %03d %%\n", 31);
```

# Opgaver

Vi mødes her igen kl. 13.15

Del 2

# Funktioner

```
type func_name(type1 arg1, type2 arg2, ...) {  
    ...  
    return ...;  
}
```

# Funktioner

```
type func_name(type1 arg1, type2 arg2, ...) {  
    ...  
    return ...;  
}
```

```
int add_one(int x) {  
    return x+1;  
}
```



# Funktioner

```
int main() {  
    printf("Hello world!\n");  
    return 0;  
}
```

# Funktioner

```
int main() {  
    printf("Hello world!\n");  
    return 0;  
}
```

```
void print_hej(void) {  
    printf("Hej!\n");  
}
```

# Funktioner

```
int main() {  
    int y = add_one(41);  
  
    return 0;  
}
```

```
int add_one(int x) {  
    return x+1;  
}
```

# Funktioner

```
int add_one(int x);

int main() {
    int y = add_one(41);

    return 0;
}

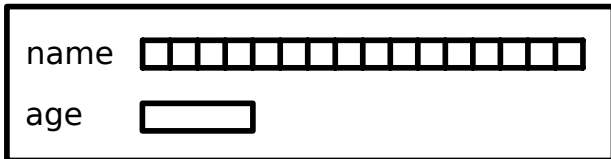
int add_one(int x) {
    return x+1;
}
```

## Strukturer ("Structs")

```
struct Person {  
    char name[16];  
    int age;  
};
```

## Strukturer ("Structs")

### Person



## Strukturer ("Structs")

```
struct Person person;
```

```
person.name[0] = 'B';
```

```
person.name[1] = 'o';
```

```
person.name[2] = 'b';
```

```
person.age     = 42;
```

## Strukturer ("Structs")

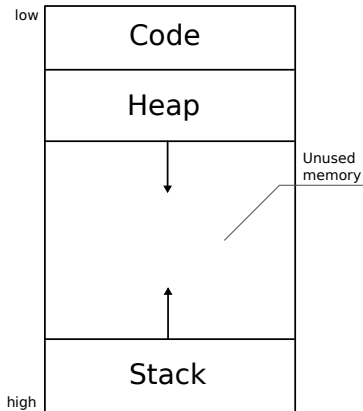
```
typedef struct Person Person;
```

```
struct Person {  
    char name[16];  
    int age;  
};
```

```
Person person;
```

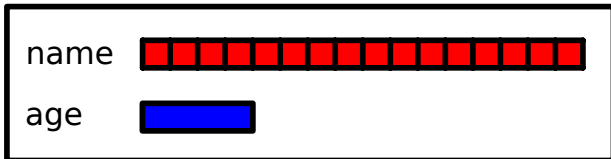


## Sidespring: Stak vs. Heap

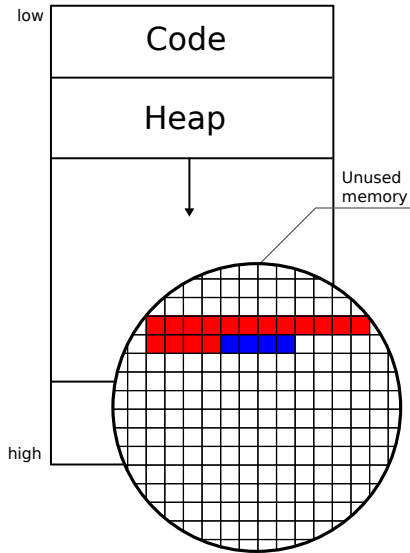


## Sidespring: Stak vs. Heap

Person



# Sidespring: Stak vs. Heap





# Pointers

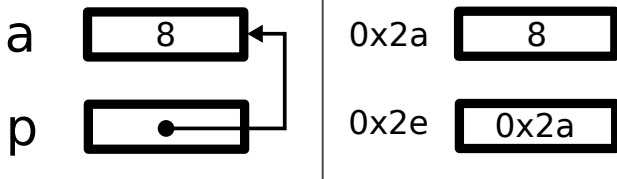
```
int a = 8;  
int* p = &a;
```

# Pointers

```
int a = 8;  
int* p = &a;
```

```
*p = 32;  
/* a is now 32 */
```

# Pointers



## Pointers & Structs

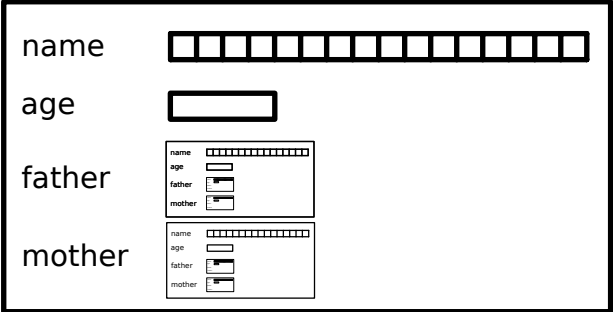
```
typedef struct Person Person;
```

```
struct Person {  
    char    name[16];  
    int     age;  
    Person  father;  
    Person  mother;  
};
```



# Pointers & Structs

## Person



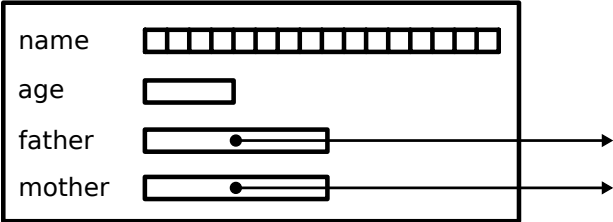
## Pointers & Structs

```
typedef struct Person Person;
```

```
struct Person {  
    char    name[16];  
    int     age;  
    Person* father;  
    Person* mother;  
};
```

# Pointers & Structs

## Person



## Pointers & Malloc

```
int* a = malloc(4);  
*a = 42;
```

## Pointers & Malloc

```
int* a = malloc(sizeof(int));  
*a = 42;
```

## Pointers & Malloc

```
Person* person = malloc(sizeof(Person));  
person.age = 42; // Doesn't work!
```

## Pointers & Malloc

```
Person* person = malloc(sizeof(Person));  
(*person).age = 42;
```

## Pointers & Malloc

```
Person* person = malloc(sizeof(Person));  
(*person).age = 42;  
person->age = 42;
```



## Pointers & Malloc

```
Person* person = malloc(sizeof(Person));
```

```
person->father = malloc(sizeof(Person));
```

```
person->father->age = 83;
```

# Pointers & Malloc

## Constructors

```
Person* create_person(int age, Person* father) {
    Person* person = malloc(sizeof(Person));
    person->age     = age;
    person->father  = father;
    return person;
}

int main() {
    Person* grandpa = create_person(83, NULL);
    Person* bob     = create_person(42, grandpa);
}
```

## Pointers & Malloc

```
Person* person = malloc(sizeof(Person));  
...  
free(person);
```

# Pointers & Malloc

## Deconstructors

```
void destroy_person(Person* person) {  
    /* Clean up other stuff */  
    ...  
    free(person);  
}
```

## Pointers & Arrays

```
int arr[16];
```

```
arr[5] = 17;
```

## Pointers & Arrays

```
int arr[16];
```

```
arr[5] = 17;
```

```
*(arr + 5) = 17;
```

## Pointers & Arrays

```
int* arr = malloc(16 * sizeof(int));
```

```
arr[5] = 17;
```

```
*(arr + 5) = 17;
```

## Pointers & Arrays

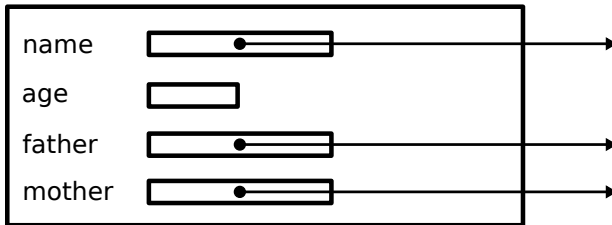
```
typedef struct Person Person;
```

```
struct Person {  
    char*   name; /* Instead of char name[16]; */  
    int     age;  
    Person* father;  
    Person* mother;  
};
```

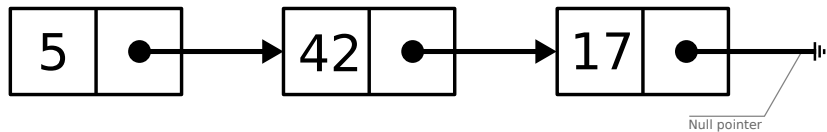


# Pointers & Arrays

Person



# Structs + Funktionen = Datastruktur



## Structs + Funktionen = Datastrukturer

```
typedef struct LinkedList LinkedList;
```

```
struct LinkedList {  
    int data;  
    LinkedList* next;  
}
```

## Structs + Funktionen = Datastrukturer

```
LinkedList* linkedlist_create(int first_element) {  
    LinkedList* ll = malloc(sizeof(LinkedList));  
    ll->data = first_element;  
    ll->next = NULL;  
    return ll;  
}
```

## Structs + Funktionen = Datastrukturer

```
LinkedList* linkedlist_insert(int element,
                               LinkedList* list) {
    LinkedList* head = malloc(sizeof(LinkedList));
    head->data = element;
    head->next = list;
    return head;
}
```

## Structs + Funktionen = Datastrukturer

```
LinkedList* ll = linkedlist_create(17);  
ll = linkedlist_insert(42, ll);  
ll = linkedlist_insert(5, ll);
```

## Input & Output

```
char name[64];  
int age;  
  
printf("What is your name?\n");  
scanf("%s", name);  
printf("Hi %s!\n", name);
```

## Input & Output

```
char name[64];
int age;

printf("What is your name?\n");
scanf("%s", name);
printf("Hi %s!\n", name);

printf("What is your age?\n");
scanf("%d", &age);
printf("Cool! My friend is also %d years old\n", age);
```



## Input & Output

```
FILE* f = fopen("names.txt", "w");  
fprintf(f, "Bob\n");
```

## Input & Output

```
FILE* f = fopen("names.txt", "r");
```

```
char c;
```

```
while ((c = fgetc(f)) != EOF) {
```

```
    printf("%c", c);
```

```
}
```

## Input & Output

```
FILE* f = fopen("names.txt", "r");

char buf[256];
while (fgets(buf, sizeof(buf), f)) {
    printf("line: %s", buf);
}
```

# Opgaver

Vi mødes her igen kl. 16.15

# Kodecafé

+ Evaluering