# Structuur van Computerprogrammas II: Week 14

Elisa Gonzalez Boix  Maarten Vandercammen  Robbe De Greef

Op Canvas vind je `oefeningen.c` met daarin code die je nodig zal hebben om deze oefeningen te maken.

## Recap-oefeningen

1. (**Dodona**) Minimaliseer het geheugenverbruik van een `Person` door een functie `uint16_t encode_person(Person p)` en een functie `Person decode_person(uint16_t)` te schrijven. `encode_person` maakt gebruik van bitmasks en bit operaties om een `Person` om te zetten naar een geëncodeerd `uint16_t` (een unsigned int dat 16 bit of 2 byte groot is). `decode_person` doet het omgekeerde.

2. (**Dodona**) Implementeer een functie `void replace_line(char *path, int line_nr, char *new_line)` die het bestand in `path` opent en lijn nummer `line_nr` (als die bestaat) volledig vervangt door `new_line`. Alle andere lijnen in dat bestand moeten hetzelfde blijven. Je mag ervan uitgaan dat het een klein bestand van minder dan 1000 karakters is en dat `new_line` altijd eindigt met het newline karakter `'\n'`. Je kan de `strcpy` uit `<string.h>` (zie `http://www.cplusplus.com/reference/cstring/strcpy/`) functie gebruiken om een string te kopiëren van de ene plaats naar de andere.

3. Hoeveel geheugen moet er minstens statisch gealloceerd worden voor de functie `read_n_people` uit het bestand `oefeningen.c`? Hoeveel geheugen wordt er dynamisch gealloceerd tijdens de executie van die functie als de gebruiker $n$ personen wil uitlezen?

4. (a) Wat zou er geprint worden als je het stuk code hieronder zou uitvoeren?

```c
int main(void) {
   int array[10];
   printf("%i", sizeof(array));
   f(array);
}

void f(int array_arg[]) {
   printf("%i", sizeof(array_arg));
```

```
}
```

(b) Wat zou er geprint worden als je de functies hieronder zou oproepen?

```
void f(int array_arg[3][5]) {
  printf("%i", sizeof(array_arg));
  printf("%i", sizeof(array_arg[0]));
}

void h(int *array_arg[3]) {
  printf("%i", sizeof(array_arg));
  printf("%i", sizeof(array_arg[0]));
}
```

(c) Zijn de volgende stukken code correct?

- ```
  int *a;
  *a = 5;
  ```
- ```
  int *a = malloc(sizeof(int));
  *a = 42;
  ```
- ```
  int array[10];
  array++;
  ```
- ```
  int array[10];
  int *a = array;
  a++;
  ```
- ```
  void f(int array[]) {
     array++;
  }

  void main(void) {
     int array[10];
     f(array)
  }
  ```
- ```
  int *a = malloc(sizeof(int));
  *a = 42;
  free(a);
  *a = 43;
  ```
- ```
  int *a = malloc(sizeof(int));int *b = a;
  *a = 42;
  free(a);
  *b = 43;
  ```
- ```
  int *a = malloc(sizeof(int));int b;
  *a = 42;
  free(a);
  a = &b;
  *a = 43;
  ```

## Extra

5. Breid de implementatie van de try/catch uit het hoorcollege uit met de mogelijkheid om geneste try's te gebruiken.