5 Structs, malloc

Oefening 32

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 4  /*1000 in de opgave*/

char* lees(){
     char tekst[MAX+1];          /* +1 !! */
     int lengte;
     char * nieuw;

     printf("Geef een tekst: ");
     fgets(tekst,MAX+1,stdin);
     lengte = strlen(tekst);

     /* indien 4+1 karaktervelden voorzien, en 'aap\n' ingelezen */
     if(tekst[lengte-1] == '\n'){
          tekst[lengte-1] = 0;    /* of '\0' */
          lengte--;
     }
     /* indien 4+1 karaktervelden voorzien, en 'noot' ingelezen:
        alles ok; niets doen of lege lus                          */
     /* indien 4+1 karaktervelden voorzien, en 'appelmoes' ingelezen:
        'appe\0' bewaard; 'lmoes' moet nog weg                    */
     else{    /* enkel als er nog iets staat!! */
          while (getchar() != '\n');
     }

     nieuw = malloc((lengte+1)*sizeof(char));
     strcpy(nieuw,tekst);
     return nieuw;
}


int main(){
     int i;
     for(i=0; i<5; i++){
          char * tekst = lees();
          printf("Ik las in %s.\n",tekst);
          free(tekst);                    /* BELANGRIJK! */
     }
     return 0;
}
```

Oefening 33

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

#define MAX 4
#define MAXAANTAL 6  /* nadien nog een nullpointer opslaan */


/* functie lees() zie vorige oefening */
```

```
char** lees_meerdere(){
    int aantal=0;
    int i;
    char* teksten[MAXAANTAL];
    char** nieuwe_teksten;

    char* tekst = lees();
    while( aantal < MAXAANTAL-1 && strcmp(tekst,"STOP") != 0 ){
        teksten[aantal] = tekst;
        aantal++;
        tekst = lees();
    }
    if(aantal == MAXAANTAL-1 && strcmp(tekst,"STOP")!=0){
        teksten[aantal] = tekst;
        aantal++;
    }
    else{
        printf("free %s\n",tekst); /* visualiseert het vrijmaken */
        free(tekst);
    }

    /* nieuwe_teksten is 1 langer (voor nullpointer)  */
    nieuwe_teksten = malloc((aantal+1)*sizeof(char*));
    i=0;
    for(i=0; i<aantal; i++){
        nieuwe_teksten[i] = teksten[i];
    }
    nieuwe_teksten[aantal] = NULL;

    return nieuwe_teksten;
}

void geef_vrij(char** teksten){
    printf("\nVrijmaken elementen: \n");
    while(*teksten){
        printf("free %s\n",*teksten); /* visualiseert het vrijmaken */
        free(*teksten);
        teksten++;
    }
}

int main(){
    char ** teksten = lees_meerdere();
    char ** w;

    printf("Ik las deze teksten in (tussen ***  ***: )\n");
    /* overloop met hulpptr; anders kan free niet meer! */
    w = teksten;
    while(*w){
        printf("***%s***\n",*w);
        w++;
    }

    geef_vrij(teksten);  /*alle elementen in de array wissen*/
    printf("\nfree **teksten"); /* visualiseert het vrijmaken */
    free(teksten);       /*zichzelf vrijgeven*/
    return 0;
}
```

Oefening 34

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

typedef struct{
    int waarde;
    int aantal_delers;
    int* delers;
}Deeltal;

void schrijf_ints(const int * x, int aantal)
{
    int i;
    printf("%i",x[0]);  /*eerste getal zonder tussenteken*/
    for(i=1; i<aantal; i++){
        printf("-%i",x[i]);
    }
}

void schrijf_deeltal(const Deeltal * x)
{
    printf("%d ",x->waarde);
    schrijf_ints(x->delers,x->aantal_delers);
    printf("\n");
}

int aantal_delers_van(int x)
{
    int aantal = 1;
    int d;
    for(d = 2; d <= x/2; d++){
        if(x%d == 0){
            aantal++;
        }
    }
    return aantal;
}

int * delers_van(int x, int aantal)
{
    int * delers = (int*) malloc(aantal*sizeof(int));
    int d, index = 0;
    for(d = 1; d <= x/2; d++){
        if(x%d == 0){
            delers[index++] = d;
        }
    }
    return delers;
}

/* g mag geen zwevende pointer zijn! */
void lees_deeltal(Deeltal * g)
{
    scanf("%i",&(g->waarde));
    if(g->waarde < 0){
        g->waarde *= -1;
    }
    g->aantal_delers = aantal_delers_van(g->waarde);
    g->delers = delers_van(g->waarde,g->aantal_delers);
}
```

```
void lees_deeltallen(Deeltal* t, int aantal)
{
    int i;
    for(i=0; i<aantal; i++){
        lees_deeltal(&t[i]);
    }
}

void schrijf_deeltallen(const Deeltal * ptr, int aantal)
{
    int i;
    for(i=0; i<aantal; i++){
        schrijf_deeltal(&ptr[i]);
    }
}

const Deeltal * zoek(int waarde, const Deeltal * t, int aantal)
{
    int i=0;
    while(i<aantal && t[i].waarde != waarde){
        i++;
    }
    return (i==aantal ? 0 : &t[i]);
}

void free_delers(Deeltal * g)
{
    /* belangrijk om te controleren of je goed bezig bent: */
    printf("\nik geef (delers van) dit deeltal vrij: %d",g->waarde);
    free(g->delers);
}

void free_deeltallen(Deeltal * g, int aantal)
{
    int i;
    printf("\nFree tabel met %d deeltallen:",aantal);
    for(i=0; i<aantal; i++){
        free_delers(&g[i]);
    }
}

void free_deeltallen_volledig(Deeltal ** g, int aantal)
{
    int i;
    printf("\nFree tabel met %d deeltallen:",aantal);
    for(i=0; i<aantal; i++){
        free_delers(&(*g)[i]);
    }
    printf("\nTabel zelf vrijgeven");
    free(*g);
}

main()
{
    Deeltal g,x;
    int t[] = {1,2,3};
    int aantal;
    Deeltal * deeltallen;
    const Deeltal * gezocht;

    g.waarde = 6;
```

```
    g.aantal_delers=3;
    g.delers = t;
    schrijf_deeltal(&g);

    printf("Geef een eerste deeltal in.\n");
    lees_deeltal(&x);
    schrijf_deeltal(&x);

    printf("Nu komt het vervolg; hoeveel deeltallen wil je nog ingeven?");
    scanf("%d",&aantal);
    printf("Geef nu die deeltallen in.\n");

    deeltallen = (Deeltal*) malloc(aantal * sizeof(Deeltal));
    lees_deeltallen(deeltallen,aantal);
    schrijf_deeltallen(deeltallen,aantal);

    printf("\nIk zoek 20: ");
    gezocht = zoek(20,deeltallen,aantal);
    if(gezocht!=0) schrijf_deeltal(gezocht);
    else printf("\n20 niet gevonden... ");

    free_delers(&g);
    free_delers(&x);

    free_deeltallen(deeltallen,aantal);
    printf("\nTabel zelf vrijgeven");
    free(deeltallen);    /* moet je nog apart doen! Tenzij je voor de alternatieve
                            procedure free_deeltallen_volledig(...) gaat :

    free_deeltallen_volledig(&deeltallen,aantal);*/

    /* merk op: 'gezocht' geef je uiteraard niet vrij,
                want dit wijst naar geheugen dat reeds vrijgegeven werd.
                (en free op een nullpointer oproepen crasht) */

    return 0;
}
```

Oefening 35

```
#include <stdio.h>
#include <string.h>
#define AANTAL_WOORDEN             10
#define GEMIDDELDE_LENGTE_WOORDEN   7
#define TOTALE_LENGTE_ARRAY  AANTAL_WOORDEN * (1+GEMIDDELDE_LENGTE_WOORDEN)

void lees(char ** pt){
    int i;
    for(i=0; i<AANTAL_WOORDEN; i++){
        scanf("%s",*pt);
        *(pt+1) = *pt+strlen(*pt)+1;
        pt++;
    }
    *pt=0;
}

void schrijf(const char * const * pt){
    while(*pt!=0){
        puts(*pt++);
    }
}
```

```
/* aanroep van de methode schrijf  in het hoofdprogramma:
     schrijf((const char * const*)pt);  /*

/********************************************** EXTRA  ************************/
/*  Merk op: als je met scanf wil aangeven hoeveel letters er maximaal ingelezen
mogen/kunnen/zullen worden, dan moet dat aantal een hardgecodeerd getal zijn.
(Zelfs het gebruik van een constante of define is niet mogelijk.)
 Bij het gebruik van fgets(..,aantal,..) hoeft dat niet;
 In deze oefening is 'aantal' een variabele, dit kan niet met scanf !          */

#include <stdio.h>
#include <string.h>
#define TOTALE_LENGTE_ARRAY    70

/* Opgelet! Bijgebruik van fgets zal de 'newline' (als gebruiker 'enter' induwt)
   ook in de c-string opgeslagen worden; dat verwijderen we eruit in regel (***) */

/*versie met indexering */
void lees1(char ** pt){
    int i = 0;
    int lengte_woord;
    int aantal_plaatsen = TOTALE_LENGTE_ARRAY;
    while(aantal_plaatsen>1){
        printf("Geef een woord in van maximaal %d karakter(s):
            ",(aantal_plaatsen-1));
        fgets(pt[i],aantal_plaatsen,stdin);
        if(pt[i][strlen(pt[i])-1] == '\n'){
            pt[i][strlen(pt[i])-1] = 0; /* (***) */
        }
        lengte_woord = strlen(pt[i]);
        pt[i+1] = pt[i] + lengte_woord+1;
        aantal_plaatsen -= lengte_woord+1;
        i++;
    }
    pt[i]=0;
}

/*versie zonder indexering */
void lees2(char ** pt){
    char **p = pt;
    int lengte_woord;
    int aantal_plaatsen = TOTALE_LENGTE_ARRAY;
    while(aantal_plaatsen>1){
        printf("Geef een woord in van maximaal %d karakter(s):
            ",(aantal_plaatsen-1));
        fgets(*p,aantal_plaatsen,stdin);
        if(*(*p+strlen(*p)-1) == '\n'){
             *(*p+strlen(*p)-1)= 0; /* (***) */
        }
        lengte_woord = strlen(*p);
        *(p+1) = *p + lengte_woord+1;
        aantal_plaatsen -= lengte_woord+1;
        p++;
    }
    *p=0;
}
/* declareer in het hoofdprogramma */

   char* pt[TOTALE_LENGTE_ARRAY/2+1];  /* zodat je ook nog een nullpointer kan
                                  wegsteken op het einde van de pointertabel */
```