4 Functiepointers, C-strings

Oefening 25

```
/* definities bovenaan is duidelijker */
void vul_tabel(const int *, const int *, int *, int, int (*)(int,int) );
int som(int , int );
int product(int , int );
int verschil(int , int );
void schrijf(const int * t, int aantal);

/* int main() was gegeven */

void vul_tabel(const int * t_1, const int * t_2, int * resultaat, int lengte, int
    (*bewerking)(int,int) ){
     int i;
     for(i=0; i<lengte; i++){
          resultaat[i] = bewerking(t_1[i],t_2[i]);
     }
}
```

Oefening 26

```
#include <stdio.h>
void my_toupper(char * s);

/* In deze oefening gebruiken we een procedure.
   In latere oefeningen wordt dit omgevormd naar een functie (nl. returntype char*)
*/

int main(){
     /* char * woord = "snEEuwwITJE<3!!";              DIT MAG NIET */
     char woord[50] = "snEEuwwITJE<3!!";     /*        DIT MAG WEL */

     /* Laatste vraag: inlezen van toetsenbord: */
     printf("Geef een woord op: ");
     scanf("%49s",woord);

     my_toupper(woord);
     puts(woord);
     return 0;
}

void my_toupper(char * s){
     /* je mag veronderstellen dat er een eerste letter is */
     if(*s >= 'a' && *s <= 'z'){
          *s = *s - 'a' + 'A';
     }
     s++;
     while(*s != 0){
          if(*s >= 'A' && *s <= 'Z'){
               *s = *s - 'A' + 'a';
          }
          s++;
     }
}
```

Oefening 27

```
/*methode schrijf vind je in oefening 21*/

const char* pointerNaarEersteKleineLetter(const char* l){
    while(*l && (*l<'a' || *l>'z')){
        l++;
    }
    return l;
}

void verzetNaarEersteHoofdletter(const char ** l){
    while(**l && (**l<'A' || **l>'Z')){
        (*l)++;
    }
}
```

Oefening 28

```
#include <stdio.h>
void wis(char * s);

int main(){
     /* mag je hier niet doen:
        char * woord = ".....";
        ZAL CRASHEN; GEEN COMPILEERFOUT    */
     char woord [] = "8d'a7!<t-)>+. -)4h&!e9)b*( )j'(e)!4\n8g|'92o!43e5d/.' 2
         3g*(e('d22a'(a25n'(";
     wis(woord);
     puts(woord);

     char woord2[81];
     printf("\n\n\nGeef nu zelf een tekst in (met spaties);\n");
     fgets(woord2,81,stdin);
     wis(woord2);
     puts(woord2);

     return 0;
}

void wis(char * s){
     char * loper = s;
     while(*s){
          if(islower(*s) || isspace(*s)){
               *loper = *s;
               loper++;
          }
          s++;
     }
     *loper = 0;
}
```

Oefening 29

```
#include <stdio.h>

/*methode to_upper(char *s) vind je in andere oplossing*/

int main(int argc,char**argv){
     int i;

     if(argc > 1) {
          for(i=1; i<argc; i++){
               my_to_upper(argv[i]);
               printf("Dag %s!\n",argv[i]);
          }

     }
     else printf("Dag allemaal!");
}
```

Oefening 30

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

/*methode to_upper(char *s) vind je in andere oplossing*/
const char* alfab_kleinste(const char* const *woorden ,int aantal);

int main(int argc ,char**argv){
     int i;
     const char * kleinste;
     for(i=1; i<argc; i++){
          my_to_upper(argv[i]);
     }
     if(argc > 1) {
          kleinste = alfab_kleinste((const char * const *)argv+1,argc -1);
          /* Merk op: sla het eerste element van argv over!
             En doe dat HIER, het is niet de taak van de functie
             alfab_kleinste om het eerste element over te slaan. */
          printf("Dag %s!",kleinste);
     }
     else printf("Dag allemaal!");
}

const char* alfab_kleinste(const char* const *woorden ,int aantal){
     const char* kl = woorden[0];
     int i;
     for(i=1; i<aantal; i++)
          if(strcmp(kl,woorden[i]) > 0)
               kl = woorden[i];
     return kl;
}
```

Oefening 31

```
#include <stdio.h>
#include <string.h>
void schrijf_int(const void * v){
     const int * a = (const int *) v;
     printf("%d",*a);
}

void schrijf_cstring(const void * v){
     const char * const * s = (const char * const *) v;
     printf("%s",*s);
}

void schrijf_double(const void * v){
     const double * x = (const double *) v;
     printf("%f",*x);
}

/************ Eerste poging  ***********/
void schrijf_array___(const void * t, int aantal, int grootte, char tussenteken,
    void (*schrijf)(const void*)){
     int i;
     schrijf(t);
     for(i=1; i < aantal; i++){
          printf("%c",tussenteken);
          t += grootte;  /* verplaats de pointer*/
          schrijf(t);
     }
     printf("\n\n");
}

/* De meeste compilers geven hier echter - terecht - warnings op:
   'pointer of type 'void*' used in arithmetic'
   Andere compilers weigeren het helemaal.
   De reden: het is niet standaard gedefinieerd wat "+i" betekent bij een
       void-pointer.

   De oplossing: de void-pointer wordt gecast naar een char-pointer, die wel
   weet hoe er gerekend moet worden. Bovendien beslaat een 'char' de kleinste eenheid
   van geheugenplaats, en zijn alle andere hier veelvoud van.
   Dus kunnen we hetzelfde principe toepassen: de stapgrootte nog aanpassen
   aan de grootte van het type waarnaar verwezen wordt.

   Met deze ingreep zijn de warnings weg, en zijn we uit de illegaliteit...  Zie ook
https://stackoverflow.com/questions/33154318/how-to-traverse-an-array-parameter-as-void-pointer.
*/

/***************** VERSIE 1 - correct ****************************************
void schrijf_array(const void * t, int aantal, int grootte, char tussenteken, void
    (*schrijf)(const void*)){
     const char* hulpptr = (const char*)t;
     int i;
     schrijf(hulpptr);
     for(i=1; i < aantal; i++){
          printf("%c",tussenteken);
          hulpptr += grootte;
          schrijf(hulpptr);
     }
     printf("\n\n");
}
```

```
int main(){
     char *namen[]= {"Evi","Jaro","Timen","Youri","Ashaf","Jennifer"};
     int leeftijden[] = {21,30,18,14,22,19};
     double scores[] = {0.5,1.6,8.2,-2.4};

     schrijf_array(leeftijden, sizeof(leeftijden)/sizeof(int) ,sizeof(int),    ',',
         schrijf_int);
     schrijf_array(namen,      sizeof(namen)/sizeof(char*)    ,sizeof(char*),  ';',
         schrijf_cstring);
     schrijf_array(scores,     sizeof(scores)/sizeof(double)  ,sizeof(double), '~',
         schrijf_double);
     return 0;
}


/* merk op:
   (1) voor schrijf_int, schrijf_cstring en schrijf_double in bovenstaande 3 regels
   mag ook een & staan, maar het moet niet (compiler kan beide interpreteren)
   (2) voor leeftijden, namen, scores in bovenstaande 3 regels mag nog (void*) staan
   (casten naar void-pointer), maar het moet niet (compiler kan hiermee weg).
*/
/*

/***************** VERSIE 2 - even correct************************************
  hier blijft alle code gelijk, behalve wat hieronder staat:                  */


void schrijf_double(const double x){
     printf("%f",*x);
}

void schrijf_int(const int * a){
     printf("%d",*a);
}

void schrijf_cstring(const char *const * s){  // !! DUBBEL STER !!
     printf("%s",*s);
}

/*aanroep in het hoofdprogramma:    */

     schrijf_array(leeftijden, sizeof(leeftijden)/sizeof(int) ,sizeof(int),    ',',
           (void (*)(const void*))schrijf_int);
     schrijf_array(namen,      sizeof(namen)/sizeof(char*)    ,sizeof(char*),  ';',
           (void (*)(const void*))schrijf_cstring);
     schrijf_array(scores,     sizeof(scores)/sizeof(double)  ,sizeof(double), '~',
       (void (*)(const void*))schrijf_double);
```