4 Functiepointers, C-strings

Oefening 25

Gegeven onderstaand hoofdprogramma, en de functies `som`, `product` en `verschil`. Je ziet dat de derde tabel wordt ingevuld aan de hand van de twee eerste tabellen. Afhankelijk van de laatste parameter van de procedure `vul_tabel`, bevat de derde tabel de som, respectievelijk product of verschil van de overeenkomstige elementen uit de eerste twee tabellen.

```
#include <stdio.h>
#define AANTAL 5
int som(int a, int b){
    return a+b;
}
int product(int a, int b){
    return a*b;
}
int verschil(int a, int b){
    return a-b;
}
void schrijf(const int * t, int aantal){
    int i;
    for(i=0; i<aantal; i++){
        printf("%i ",t[i]);
    }
    printf("\n");
}
```

```
int main(){
    int a[AANTAL];
    int b[AANTAL];
    int c[AANTAL];
    int i;
    for(i=0; i<AANTAL; i++){
        a[i] = 10*i;
        b[i] = i;
    }

    vul_tabel(a,b,c,AANTAL,som);
    schrijf(c,AANTAL);

    vul_tabel(a,b,c,AANTAL,product);
    schrijf(c,AANTAL);

    vul_tabel(a,b,c,AANTAL,verschil);
    schrijf(c,AANTAL);
    return 0;
}
```

Als output zal er dus verschijnen:

```
0 11 22 33 44
0 10 40 90 160
0 9 18 27 36
```

Schrijf de procedure `vul_tabel(...)`. De laatste parameter is een functie. Definieer de procedure `vul_tabel` ná je `main`-functie. Dan moet je `vul_tabel` vooraf declareren. Laat in die parameterlijst de benamingen van de parameters zelf weg, schrijf enkel de types neer.

Oefening 26

Schrijf een procedure `my_toupper(s)` die de eerste letter van de gegeven C-string `s` omzet naar een hoofdletter, en de andere letters naar een kleine letter. Cijfertekens en leestekens worden niet beïnvloed. Je mag er vanuit gaan dat het eerste karakter een letter is. Gebruik geen functies uit de cstring-bibliotheek, maar doe de aanpassingen zelf. Gebruik schuivende pointers.

Test uit met een hoofdprogramma waarin je het woord `snEEuwwITJE<3!!` laat omzetten naar `Sneeuwwitje<3!!`. Daarna test je ook uit met een woord dat je inleest van het toetsenbord.

Oefening 27

Gegeven onderstaand hoofdprogramma. Schrijf de nodige code om dit te laten werken; gebruik overal **verplicht schuivende pointers**. De procedure `schrijf(begin,eind)` werd al gevraagd in oefening 21.

1. de functie `pointerNaarEersteKleineLetter(p)` geeft een pointer terug naar de eerste kleine letter die te vinden is vanaf de huidige positie van de pointer `p`. Indien er geen kleine letters meer volgen, staat `p` op het einde van de c-string (=voorbij de laatste letter).
2. de procedure `verzetNaarEersteHoofdletter(p)` verzet de gegeven pointer `p` zodat hij wijst naar de eerste hoofdletter die vanaf zijn huidige positie te vinden is. (Ook hier: voorzie de situatie waarbij er geen hoofdletters meer volgen.)

```
int main(){
    const char zus1[50] = "sneeuwWITje";
    const char zus2[50] = "rozeROOD";
    const char* begin;
    const char* eind;
    begin = zus1;
    verzetNaarEersteHoofdletter(&begin);
    eind = pointerNaarEersteKleineLetter(begin);
    schrijf(begin,eind);   /* schrijft 'WIT' uit */
    printf("\n");
    begin = zus2;
    verzetNaarEersteHoofdletter(&begin);
    eind = pointerNaarEersteKleineLetter(begin);
    schrijf(begin,eind);   /* schrijft 'ROOD' uit */
    return 0;
}
```

Oefening 28

Schrijf een procedure `wis(s)` die uit de string `s` alle tekens wist die geen kleine letter of geen white space zijn. Je mag gebruik maken van de functies `islower(char)` en `isspace(char)`. Test uit met de string
`"8d'a7!<t-)>+. -)4h&!e9)b*( )j'(e)!4\n8g|'92o!43e5d/.' 2 3g*(e('d22a'(a25n'("`.

Test ook uit met een zinnetje dat je inleest (gebruik spaties in je input).

Oefening 29

Schrijf een programma dat je bewaart onder de naam `begroet.c`. **Op de commandolijn** zal je, na de naam van het programma, een aantal persoonsnamen opgeven. Het programma antwoordt dan met een groet aan alle vermelde personen. Elke naam moet met een hoofdletter starten en omgezet worden zoals in oefening 26 (hergebruik code). Voorbeeld:

Geef op de commandolijn: `begroet oriana thomas han`

Dan komt er als reply:

```
Dag Oriana!
Dag Thomas!
Dag Han!
```

Worden er geen namen meegegeven, dan komt er `Dag allemaal!`.

Oefening 30

Werk verder op vorige oefening. Indien het programma op de commandolijn wordt opgeroepen zonder parameters, dan komt er nog altijd `Dag allemaal!`. In het andere geval begroet je enkel de persoon waarvan de naam alfabetisch het eerst komt. Dat zou in het voorbeeld van vorige opgave dus `Han` zijn.
Schrijf de functie `alfab_kleinste(voornamen,n)` die een pointer teruggeeft die wijst naar de naam die alfabetisch eerst staat in de array van c-strings (met naam `voornamen`) die als parameter wordt meegegeven. De tweede parameter is de lengte `n` van de array.

Oefening 31

Intialiseer in het hoofdprogramma volgende arrays:

```
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};
```

We willen de elementen van deze arrays uitschrijven, gescheiden door een leesteken naar keuze. Omdat de arrays elk van een ander type zijn, zouden we drie keer bijna dezelfde code moeten schrijven: een lus om de elementen van een array te overlopen en die uit te schrijven.

Dat dubbel werk kunnen we vermijden. Ga daarvoor als volgt te werk:

1. Schrijf drie procedures `schrijf_cstring`, `schrijf_int` en `schrijf_double` die alledrie één parameter van een pointertype meekrijgen en één element uitschrijven. Test uit.
2. Schrijf de procedure

   ```
   void schrijf_array(const void * t, int aantal, int grootte, char tussenteken,
                      void (*schrijf)(const void*))
   ```

   die de eerste `aantal` elementen uit de array `t` uitschrijft. Elk koppel elementen wordt gescheiden door het opgegeven `tussenteken`. De parameter `grootte` bevat de grootte van het type dat uitgeschreven moet worden. De parameter `schrijf` is een pointer naar een passende uitschrijfprocedure.

Schrijf een hoofdprogramma dat volgende output produceert:

```
21,30,18,14,22,19

Evi;Jaro;Timen;Youri;Ashaf;Jennifer

0.5~1.6~8.2~-2.4
```

Belangrijke voetnoot met het oog op de labotest. Zorg dat je deze oefening helemaal binnenste buiten draait en jezelf grondig bevraagt over het hoe en waarom van welke parametertypes, sterretjes, haakjes,... Een dergelijke vraag komt dikwijls op testen voor, waarbij je soms ook zelf de hoofding van de procedure moet schrijven.