3 pointers

Oefening 18

```
/* 1*/  i = j;   /* zinloos: i had een degelijke waarde; j bevat 'rommel'.
   na deze actie weet je niet meer wat in i zit. Voor het
   vlot vervolg van de oefening negeren we deze regel!                    */
/* TIP: lees  '='  als 'krijgt de waarde van'                             */
/* 2*/ jp = &i; /* ok: jp wijst naar geheugenplaats met naam i; types kloppen */
/* 3*/ j = *jp; /* ok: j krijgt nu de waarde 7                            */
/* 4*/ *ip = i; /* FOUT: ip is zwevende pointer;
                        *ip is geen gereserveerde geheugenplaats          */
/* 5*/ ip = jp; /* ok: ip wijst nu ook naar geheugenplaats met naam i     */
/* 6*/ &i = ip; /* FOUT: een adres kan je niet wijzigen                   */

/* 7*/ (*ip)++; /* ok: 7 wordt nu 8                                       */
/* 8*/ *ip *=i; /* ok: 8 wordt nu 64                                      */
/* 9*/ ip++; /* zinloos: ip wijst niet naar array; buur van ip is onbekend */

/*10*/ tp = t+2;          /* ok: tp wijst nu naar derde element in de array t  */
/*11*/ j = &t[5] - tp;    /* ok: j wordt nu 5-2=3                              */
/*12*/ t++;               /* FOUT: array kan je niet opschuiven                */
/*13*/ (*t)++;            /* ok: eerste element van de tabel verhoogt met 1    */
/*14*/ *tp--;             /* zeer slordig; not done:
  eerst vraag je "*tp" op (onzin als het alleen op de regel staat),
  en dan schuif je pointer 1 naar voor (dat op zich kan wel)                */
/*15*/ j = (*tp)++;  /* ok: aangenomen dat tp in vorige regel een plaats naar
  links opschoof in de array, krijgt j nu de waarde van t[1]- waarna dat
  element met 1 verhoogt
  (concreet: j is nu 20, terwijl de array 10 21 30 40 ... bevat)            */
/*16*/ i = *tp++;     /* ok: i krijgt de waarde van t[1] (concreet: 21),
  tp schuift een plaats naar rechts in de tabel (concreet: tp wijst naar t[2])*/

/*17*/ p1 = ip;       /* ok: p1 is pointer naar const en belooft dus meer
  dan ip deed                                                               */
/*18*/ jp = p1;       /* geeft FOUT of warning en is ZEKER NOT DONE:
  p1 is pointer naar const, terwijl jp niet belooft om de geheugenplaats
  waarnaar-ie wijst  ongewijzigd te laten.                                  */
/* warning: "assignment discards 'const' qualifier from
   pointer target type [enabled by default]"                                */
/*19*/ (*p1)--;       /* FOUT: p1 is een pointer naar const, dus
  mag je via p1 geen wijzigingen aanbrengen aan *p1                         */
/*20*/ dp = &i;       /* geeft FOUT of warning: pointer naar double
  kan niet geinitialiseerd worden met adres van int                         */
/*21*/ dp = ip;       /* idem als hiervoor                                 */

/*22*/ jp = p2;       /* ok: jp wijst nu naar element waar p2 naar wijst
  (p2 is een constante pointer dus verhuist zelf nooit, maar heeft niet beloofd
  om de geheugenplaats  waar-ie naar wijst ongewijzigd te laten, dus moet
  jp dat ook niet doen)                                                     */
/*23*/ p2 = p1;       /* FOUT: p2 is een constante  pointer,
  en kan dus geen nieuwe waarde aannemen                                    */
/*24*/ *p2 += i;      /* in orde; gezien p2 bij initialisatie naar i verwees
  (regel 23 negeren we), wordt de inhoud van i bij deze verdubbeld          */
```

Oefening 19

```
/* Oplossing:  op het scherm verschijnt er

0 11 20 62 40 50
11 62 62

Let op voor de lijn
    (*ppt)++;   /* *ppt is een pointer - die wordt een plaats naar rechts
       opgeschoven */

Oplossing van de bijvraag:

In de eerste en laatste for-lus kan je de bovengrens 3 vervangen door
   sizeof(pt)/sizeof(int*).
In de tweede for-lus kan je de bovengrens 6 vervangen door sizeof(t)/sizeof(int).
*/
```

Oefening 20

```
void wissel(int * a, int * b){
     printf("  Bij start van de wisselprocedure hebben we a=%i en b=%i.\n",*a,*b);
     int hulp = *a;
     *a = *b;
     *b = hulp;
     printf("  Op het einde van de wisselprocedure hebben we a=%i en b=%i.\n",*a,*b);
}

int main(){
    int x, y;
    x = 5;
    y = 10;

    printf("Eerst hebben we x=%i en y=%i.\n",x,y);
    wissel(&x,&y);
    printf("Na de wissel hebben we x=%i en y=%i.\n",x,y);
    return 0;
}
```

Oefening 21

```
void schrijf_aantal(const char a[], int n){  /* of (const char *a, int n)  */
     int i;
     for(i=0 ; i<n ; i++){
          printf("%c",a[i]);
     }
}

void schrijf(const char* begin, const char* eind){
     while(begin < eind){
          printf("%c",*begin);
          begin++;
/* de twee statements kunnen ook in 1 opdracht: printf("%c",*begin++);    */
     }
}

   /* voeg onderstaande code toe in het hoofdprogramma: */
     printf("Aantal elementen in array   :      %d\n",sizeof(letters)/sizeof(char));
```

```
     printf("Aantal elementen via pointer:      %d\n",sizeof(p)/sizeof(char));

   /* sizeof pointer is altijd 8 !*/
     printf("sizeof pointer to letter %d: \n", sizeof(p));
     printf("sizeof char*             %d: \n", sizeof(char*));
     printf("sizeof int*              %d: \n", sizeof(int*));
     printf("sizeof double*           %d: \n", sizeof(double*));
     printf("sizeof void*             %d: \n", sizeof(void*));
```

Oefening 22

```
#include <stdio.h>

void zoek_extremen(const int  rij[], int lengte, int * min, int * max){
     int i;

     *min = rij[0];
     *max = rij[0];

     for(i=1; i<lengte; i++){
          if(*min > rij[i]){
               *min = rij[i];
          }
          else if(*max < rij[i]){
               *max = rij[i];
          }
     }
}

void zoek_extremen_rec(const int rij[], int lengte, int * min, int * max){
     if(lengte == 1){
          *min = rij[0];
          *max = rij[0];
     }
     else{
          zoek_extremen_rec(rij,lengte-1,min,max);
          if(rij[lengte-1] < *min){
               *min = rij[lengte-1];
          }
          if(rij[lengte-1] > *max){
               *max = rij[lengte-1];
          }
     }
}

int main(){
     int rij [] = {5,7,9,6,4,2,3,8,5,-11,999,-11,5,4,2};
     /* int rij [] = {-5,-2,-888,-1,-3,-9,-4,-8,-7};       */

     int min,max;
     /* zoek_extremen(rij,sizeof(rij)/sizeof(int),&min,&max); */
     zoek_extremen_rec(rij,sizeof(rij)/sizeof(int),&min,&max);

     printf("min is %d,  max is %d",min,max);
     return(0);
}
/*
 *  Belangrijk: test ook met enkel negatieve getallen!
 *  Heb je opgemerkt dat de initialisatie van zowel min als max gebeurt met het
 *  EERSTE ELEMENT uit de array, in plaats van met de willekeurig gekozen waarde 0?
 */
```

Oefening 23

```
#include <stdio.h>
/* In deel 2 van deze vraag wordt de returnwaarde van deze functie gebruikt
   om het gevonden getal te bewerken! Dit kan niet indien het returntype const int*
       gebruikt wordt.
   Nu moet ook 'const' bij de eerste parameter weggehaald worden:
   als we een niet -const pointer willen teruggeven naar een array, dan moeten we
       toegang
   hebben tot deze array aan de hand van een niet -const pointer! */

int* plaats_van(int array[], int aantal , int gezocht) {
     int i = 0;
     while(i<aantal && gezocht != array[i])
          i++;
     return (i<aantal ? &array[i] : NULL);
}
/* alternatieve oplossing:
   int* plaats_van(int array[], int aantal , int gezocht) {
   int i;
   for (i=0 ; i<aantal ; i++)
   if (gezocht == array[i])
   return &array[i];
   return 0;
   }
*/
void plaats_ptr_op_getal(int ** dptr, int aantal, int gezocht) {
     int i = 0;
     while(i<aantal && gezocht != **dptr) {
          (*dptr)++;
          i++;
     }
     if (i==aantal)
          *dptr = NULL;
}
/* Merk op: om snel te controleren zonder inmenging van de gebruiker,
   kan je de waarde van x hardcoderen. Dat kan tijd sparen bij het testen. */
int main() {
     int rij []= {8,4,2,0,6,10};
     int lengte = sizeof(rij)/sizeof(rij[0]);
     int x, i;
     int *plaats, *ptr;
     printf("Geef een geheel getal op ");
     scanf("%d",&x);
     plaats = plaats_van(rij,lengte ,x);
     if(plaats == NULL) {
          printf("\nHet getal %d werd niet gevonden in de niet-geordende array.",x);
     }
     else {
          printf("\nIk vond het getal en vervang het door zijn tweevoud.\n");
          *plaats *= 2;
          for(i=0; i<lengte; i++)
               printf("%d ",rij[i]);
     }
     printf("\n\nVIERDE DEEL VAN DE OEFENING\n\nGeef een geheel getal op ");
     scanf("%d",&x);
     printf("\nJe gaf het getal %d op.",x);
     ptr = rij; /*zet de pointer klaar vooraan in de array */
     plaats_ptr_op_getal(&ptr,lengte ,x);
     if(ptr == NULL)
          printf("\nHet getal %d werd niet gevonden in de niet -geordende array.",x);
```

```
     else {
          printf("\nIk vond het getal en print de array vanaf dat getal:\n");
          for(i=0; i<lengte - (ptr-rij); i++)
               printf("%d ",ptr[i]);
     }
     return 0;
}
```

Oefening 24

```
#include <stdio.h>

void pivoteer( char * begin, char * einde , char * pivot){
     char *p,*q;
     p = pivot-1;
     q = pivot+1;
     while (p>=begin && q<einde){
          char h = *p;
          *p = *q;
          *q = h;
          p--;
          q++;
     }
}
```