3 pointers

Oefening 18

Beantwoord deze vraag zonder computer. Welke regels zijn syntactisch niet correct - en zullen een compilerfout opleveren. Welke bewerkingen kloppen wel qua syntax, maar zijn zinloos en/of gevaarlijk voor het verloop van het programma.
Wat zit er uiteindelijk in de variabelen? Regels die een compilefout geven, of die zinloos/gevaarlijk zijn negeer je hierbij.

```
int main(){
    int i=7, j;
    double d;
    int *ip, *jp, *tp;
    double *dp;
    const int * p1;
    int * const p2 = &i;
    int t[25] = {10,20,30,40,50,60};

    /* 1*/ i = j;
    /* 2*/ jp = &i;
    /* 3*/ j = *jp;
    /* 4*/ *ip = i;
    /* 5*/ ip = jp;
    /* 6*/ &i = ip;

    /* 7*/ (*ip)++;
    /* 8*/ *ip *=i;
    /* 9*/ ip++;

    /*10*/ tp = t+2;
    /*11*/ j = &t[5] - tp;
    /*12*/ t++;
    /*13*/ (*t)++;
    /*14*/ *tp--;
    /*15*/ j = (*tp)++;
    /*16*/ i = *tp++;

    /*17*/ p1 = ip;
    /*18*/ jp = p1;
    /*19*/ (*p1)--;
    /*20*/ dp = &i;
    /*21*/ dp = ip;

    /*22*/ jp = p2;
    /*23*/ p2 = p1;
    /*24*/ *p2 += i;
    return 0;
}
```

Oefening 19

Wat wordt er uitgeschreven? Uiteraard beantwoord je deze vraag zonder computer.
Bijvraag: hoe zou je de hardgecodeerde bovengrenzen van de drie for-lussen kunnen vervangen door zelf de grootte van de array's te bepalen?

```
#include <stdio.h>
int main(){
    int t[6] = {0,10,20,30,40,50};
    int* pt[3];

    int i;
    for(i=0; i<3; i++){
        pt[i] = &t[2*i];
    }

    pt[1]++;
    pt[2] = pt[1];
    *pt[1] += 1;
    *pt[2] *= 2;
    int ** ppt = &pt[0];
    (*ppt)++;
    **ppt += 1;

    for(i=0; i<6; i++){
        printf("%d ",t[i]);
    }
    printf("\n");
    for(i=0; i<3; i++){
        printf("%d ",*pt[i]);
    }
    printf("\n");
    return 0;
}
```

Oefening 20

Keer terug naar oefening 11. Hier werd een `wissel`-procedure gegeven, die echter niet deed wat ze beloofde. Pas de code aan (haal eerst het bestand `opg_19_11_wissel.tex` af van Minerva (map met LaTeX-bestanden (extensie `.tex`)), zodat je geen code hoeft over te tikken). Aan het hoofdprogramma verander je niets of zo weinig mogelijk.

Oefening 21

Dit is een vervolg op oefening 13. Gegeven onderstaand hoofdprogramma:

```
#include <stdio.h>
int main(){

    char letters [] = {'p','o','r','g','o','e','d','o','i','e','o','k',
                       'i',':','a','-','t','('};
    const char *p = letters;

    schrijf_aantal(letters+3,4);
    printf("\n");
    schrijf(p+10,p+12);

}
```

1. Kan je de lengte van de array opvragen via de pointer `p`?
   Ga na dat de geheugenruimte, ingenomen door een pointer altijd gelijk is (onafhankelijk van het type waarnaar de pointer verwijst). niet gelijk!!!!
2. Schrijf de procedure `schrijf_aantal` die het aantal gevraagde letters uitschrijft vanaf de pointer opgegeven in de eerste parameter.
3. Schrijf de procedure `schrijf(begin,eind)` die de letters uitschrijft die te vinden zijn tussen de plaats waar de pointers `begin` en `eind` naar wijzen. (Laatste grens niet inbegrepen; je mag er vanuit gaan dat beide pointers wijzen naar elementen in dezelfde array van karakters, en dat `begin < eind` .)

Oefening 22

- Schrijf een procedure `zoek_extremen(...)` die op zoek gaat naar het minimum én maximum in een array van gehele getallen. Het type en de aard van de parameters bepaal je zelf. In het hoofdprogramma zorg je voor een hardgecodeerde array, en je schrijft het minimum en maximum uit. Test kritisch!
- Schrijf een recursieve versie van deze procedure.

Oefening 23

Zorg dat je oefening 14 onder de knie hebt, voor je hieraan begint. In deze oefening wordt met gehele getallen gewerkt in plaats van reële getallen.

1. Schrijf een functie `plaats_van(...)`, die de (meest linkse) plaats teruggeeft waarop een gegeven geheel getal in een gegeven array van gehele getallen gevonden wordt. Bepaal zelf aantal en type van de parameters. Indien het getal niet aanwezig is, wordt de nullpointer teruggegeven.
2. Schrijf een hoofdprogramma om je functie uit te testen. Probeer alle randgevallen: zoek het eerste getal, het laatste getal, en een getal dat niet in de array voorkomt.
3. Schrijf een procedure `plaats_ptr_op_getal(...)`, die een gegeven pointer verplaatst naar de plaats waarop een gegeven geheel getal in een gegeven array van gehele getallen gevonden wordt. Indien het getal niet aanwezig is, wordt de gegeven pointer de nullpointer. Gebruik de functie `plaats_van` *niet*; gebruik schuivende pointers in plaats van indexering.
4. Breid je hoofdprogramma uit: verander één van de getallen uit de gegeven array in zijn tweevoud, door eerst de procedure `plaats_ptr_op_getal` op te roepen en dan het gevonden getal (in de array!) te verdubbelen. Je zal eventueel de types van de parameter en het returntype van de functie `plaats_van(...)` moeten aanpassen!

   Schrijf ter controle de hele array uit. (Wat zou er gebeuren als je een getal dat niet aanwezig is in de array wil verdubbelen? Voorziet jouw code dat er dan een foutboodschap komt in plaats van een crash?)

Oefening 24

Gegeven onderstaand hoofdprogramma hieronder. Gebruik de functie `schrijf(begin,einde)` uit oefening 21)

```
main(){
    char tekst[] = {'b','d','?','z','g','o','e','z','e','b',
                    ' ','d','i','g','!','h','o','s','v'};
    pivoteer(tekst+7,tekst+12,tekst+9);
    schrijf(tekst+4,tekst+15);
}
```

Schrijf de procedure `pivoteer(begin,einde,pivot)`. De drie parameters zijn pointers naar karakters in dezelfde array. Je mag veronderstellen dat `begin < pivot < einde`.

De procedure wisselt de elementen symmetrisch rond de pivot. Ligt de pivot niet netjes in het midden tussen de grenzen `begin` en `einde`, dan wordt het wisselen beperkt.

Een voorbeeld: indien de elementen vanaf `begin` tot net voor `einde` gelijk zijn aan `a b c d e f g h` en `pivot` wijst naar de letter `c`, dan wordt dit rijtje na afloop van de procedure `e d c b a f g h`.

Gebruik schuivende pointers.

Is de uitvoer van je programma geen leesbare uitspraak, dan schort er nog wat aan.