

Tentamen Programmeermethoden

Vrijdag 20 april 2001, 9.00–12.00 uur

Universiteit Leiden — Informatica

Bij alle te schrijven functies moeten de variabelen in de heading voorkomen (niet stiekem globale variabelen gebruiken). De opgaven tellen alle vier even zwaar mee. Veel succes!

1. We hebben een array `A` (`double A[n]`, met `const int n = 100000;`) met n verschillende getallen.

a. Geef een `int` C++-functie `grokle (A,t,gr)` die de *array-index* van het grootste getal uit de eerste t ($t \leq n$) getallen uit `A` teruggeeft (als de boolean `gr true` is), en anders de *array-index* van het kleinste getal uit de eerste t .

b. Geef een C++-functie `void wissel (A,i,j)` die de waarden van de array-elementen `A[i]` en `A[j]` verwisselt ($0 \leq i, j \leq n-1$).

c. Geef een C++-functie `void sorteer (A,n)` die het array `A` opend sortert, door herhaald het grootste getal op de juiste plaats te zetten. Het array mag alleen benaderd worden via de functies `grokle` en `wissel`.

d. Hoeveel vergelijkingen tussen `double`'s doet het algoritme van `c` in totaal? Geef een antwoord voor algemene n ; in het antwoord mag eventueel een sommatie blijven staan.

e. Als je grootste en kleinste getal uit het gehele array `A` wilt vinden, en daarvoor twee keer de functie van `a` gebruikt, hoeveel vergelijkingen tussen `double`'s kost dat dan?

f. Stel dat je het probleem van `e` als volgt zou oplossen. Onthoud twee kandidaten `x` en `y` voor grootste en kleinste; vergelijk steeds twee opeenvolgende array-elementen, en vergelijk de grootste van deze twee met `x` en de kleinste met `y`, en update deze zondig. Hoeveel vergelijkingen tussen `double`'s kost dit?

2. a. Bij een functie kun je te maken hebben met *call by value* en *call by reference*, en ook met *locale* en *globale* variabelen. Verder heb je ook nog *formele* en *actuele* parameters. Leg deze zes begrippen aan de hand van een *klein* voorbeeld duidelijk uit.

b. Een zeker C++-programma bevat de volgende declaraties:

```
double aaa (double& r, double& s) {
    r = s - r; s = s - r; r = r + s; return r - s; } // aaa
double bbb (double& r, double& s) {
    double temp = 7.0; int i = 4;
    for ( i = 0; i < 4; i++ ) {
        temp += aaa (s,r); cout << i << r << s << temp << endl; } // for
    return temp; } // bbb
```

Wat levert op (met `x` en `y` van type `double`):

```
x = 1.0; y = 3.0; cout << bbb (x,y) << x << y << endl;
```

Wat wordt er afgedrukt? Geef hierbij uiteraard uitleg.

c. Idem, maar nu zonder de vier `&`'s in de headings van `aaa` en `bbb`.

d. Wat levert op (maar nu weer met `&`-s voor de parameters in `aaa` en `bbb`):

```
x = 4.0; cout << bbb (x,x) << x << endl;
```

e. Geef een veel eenvoudiger functie `bbb2` die dezelfde return-waarde oplevert als `bbb` voor alle parameters `r` en `s`, zowel voor het geval *met* als het geval *zonder* de vier `&`'s.

3. We hebben een m bij n array A met gehele getallen (voorbeeld met $m = 2$ en $n = 5$):

```
3 3 7 2 1
3 3 8 3 5
```

a. Schrijf een C++-functie `int hoe1vaak (A,m,X)` die bepaalt hoe vaak het gehele getal X in het m bij n array A voorkomt.

b. Idem `int hoe2vaak (A,m,X,i,j)`, maar nu met uitzondering van de getallen uit de i -de rij en de getallen uit de j -de kolom. In het voorbeeld komt (met $i = 1$ en $j = 0$ het getal 3 nog één keer voor).

c. Schrijf een `int` C++-functie `uniek (A,m,i,j)` die een rijindex i en een kolomindex j oplevert (call by reference variabelen!) met de volgende eigenschap: als je de getallen uit de i -de rij en de getallen uit de j -de kolom weglaat, komt ieder getal nog precies één keer voor. Als er meerdere mogelijkheden voor i en j zijn, mag er hieruit één willekeurig gekozen worden. In het voorbeeld: $i = 1$ en $j = 0$, of $i = 1$ en $j = 1$. Als er geen mogelijkheden zijn moeten ze beide -1 worden.

4. Gegeven is het volgende type:

```
class vakje { public: // een struct mag ook
    int info; vakje* nogeen; vakje* volgende;
}; // vakje
```

Met behulp hiervan kunnen rijtjes met getallen erin worden opgebouwd. Het veld `volgende` bevat steeds een pointer naar het er direct “rechts” naast gelegen vakje of eventueel `NULL`—bij het laatste vakje. Het veld `nogeen` bevat een pointer naar een vakje, bijvoorbeeld in een andere rij. Een voorbeeld met twee rijen, met ingangen `ingang1` en `ingang2` van type `vakje*`:

```
ingang1 → 3   → 2   → 1   NULL
           ↑↓   ↑↓   ↑↓
ingang2 → 8   → 3   → 5   NULL
```

a. Schrijf een C++-functie `voeg1toe (ingang,nummer)` die een nieuw vakje met `nummer` in het `info`-veld vooraan de structuur (met `ingang` van type `vakje*` als `ingang`) toevoegt. Het veld `nogeen` van het nieuwe vakje moet `NULL` gemaakt worden.

b. Schrijf een C++-functie `voeg2toe (ingang1,nummer1,ingang2,nummer2)` die twee nieuwe vakjes met `nummer1` respectievelijk `nummer2` erin vooraan twee rijen toevoegt, waarbij de ingangen zijn als in het voorbeeld (daar zijn zojuist 3 en 8 zo voorgevoegd). De `nogeen`-velden van de nieuwe vakjes moeten naar elkaar gaan verwijzen. Gebruik **a**.

c. Schrijf een C++-functie `verwijder (ingang)` die het eerste vakje voor uit de lijst (met `ingang` van type `vakje*` als `ingang`) verwijdert indien in dat vakje het `nogeen`-veld `NULL` is of een vakje aanwijst met een grotere waarde in het `info`-veld. Denk aan de lege lijst.

d. Schrijf een boolese C++-functie `gelijk (ingang1,ingang2)` die `true` teruggeeft precies dan als de lijsten met `ingang1` en `ingang2` exact dezelfde getallen (in dezelfde volgorde) bevatten—en anders `false`.

e. In de functies bij **a**, **b**, **c** en **d** staan in de heading de parameters `ingang`, `ingang1` en `ingang2`. Deze heb je call by value of call by reference doorgegeven (met een `&`). Maakt het voor de werking van deze functies verschil uit of die `&` erbij staat? Leg ook uit wat er bij deze twee situaties precies gebeurt tijdens executie van de betreffende functie.

Zodra het tentamen is nagekeken wordt dit vermeld op het bord in de hal, hopelijk rond 7 mei.