Tentamen Inleiding Programmeren voor LS&Ters
Vrijdag 28 juni 2002, 14.00–17.00 uur
Universiteit Leiden — Informatica

De opgaven tellen alle vier even zwaar mee. Veel succes!

1. **a.** Schrijf een C++-functie `double gem2 (double x, double y)` die het gemiddelde van de twee getallen `x` en `y` teruggeeft (met behulp van een `return`-statement).
b. Schrijf een C++-functie `double gem4 (double x, double y, double u, double v)` die het gemiddelde van de vier getallen `x`, `y`, `u` en `v` teruggeeft. De functie moet gebruik maken van de functie `gem2` van **a**; het kan in één regel.
c. Schrijf een C++-functie `int wortel (double x)` die de wortel uit het positieve getal `x` als volgt benadert. Het grootste gehele getal waarvan het kwadraat nog kleiner dan of gelijk aan `x` is moet worden geretourneerd. Gebruik een `while`-loop.
d. Stel we hebben een array `woordje` met `5` letters (`char woordje[5]`) en een array `verhaal` met `100` letters (`char verhaal[100]`). We willen weten of het `woordje` in het `verhaal` voorkomt: op `5` direct opeenvolgende posities moeten de twee arrays precies overeenstemmen. Schrijf een boolese C++-functie die dit controleert.

2. We hebben een array `A` (`double A[n]`, met `const int n = 1234;`) met `n` *verschillende* getallen.
a. Geef een C++-functie `void wissel (double A[n], int i, int j)` die de waardes van de array-elementen `A[i]` en `A[j]` verwisselt (bij vaste `i` en `j` met $0 \leq$ `i`, `j` $\leq$ `n-1`).
b. Geef een C++-functie `int grootste (double A[n], int i, int j)` die het grootste getal uit `A[i]`, `A[i+1]`, ..., `A[j]` opzoekt, en de desbetreffende array-index teruggeeft.
c. Schrijf een C++-functie `void sorteer (double A[n])` die het array `A` oplopend sorteert door herhaald de functies van **a** en **b** aan te roepen.
d. Leg kort in woorden uit hoe *bubblesort* werkt.

3. **a.** Bij een functie kun je te maken hebben met *call by value* en *call by reference*, en ook met *locale* en *globale* variabelen. Leg deze vier begrippen in woorden kort en duidelijk uit.
b. Een zeker C++-programma bevat de volgende programmaregels:

```
int brom (int & x) {
  x = x + 5; return x;
} // brom
void hmm (int a, int b, int & c ) {
  int z = 6;
  c = a + b + brom (z);
  b = a + b; a = b - a; b = b - a;
  z++;
  cout << a << b << c << z << endl;
}  // hmm
```

Wat levert op (met `x`, `y` en `z` van type `int`;):

```
x = 1; y = 3; z = 5; hmm (x+y,y-x,z); cout << x << y << z << endl;
```

Wat wordt er afgedrukt? Geef hierbij uiteraard uitleg.
c. Idem, maar nu zonder de `&` in de heading van `brom` en zonder die in de heading van `hmm` (beide dus weggelaten).
d. Wat levert op (maar nu met `&`-s voor *alle* parameters in `brom` en `hmm` (4 stuks)):

```
x = 1; y = 3; z = 5; hmm (x,x,x); cout << x << y << z << endl;
```

e. Mag ergens in het programma de aanroep `brom (brom (y))` voorkomen? Leg uit.

4. Deze som gaat over een *afstandentabel*. Deze zit opgeslagen in een 2-dimensionaal array `int A[n][n];` met `n` rijen en `n` kolommen. Een voorbeeld met `n = 3`:

```
 0  3 18
 3  0  7
18  7  0
```

Het getal in de `i`-de rij, `j`-de kolom geeft de afstand aan tussen de plaatsen `i` en `j`: de afstand tussen `0` en `2` is `18`.

a. Schrijf een C++-functie `void verste (int A[n][n], int& i, int& j)` die de twee het verst uit elkaar gelegen plaatsen opzoekt in het array `A` en deze in de call-by-reference parameters `i` en `j` stopt, met in `i` het kleinste rangnummer. In het voorbeeld zijn het `0` en `2` (afstand `18`).
b. Normaal is het zo dat als je rechtstreeks van `i` naar `j` reist, dit korter is dan als je via `k` gaat. Schrijf een C++-functie `void driehoek (int A[n][n])`, die controleert of dit voor alle drietallen plaatsen `i`, `j` en `k` in `A` klopt. Het resultaat wordt op het scherm afgedrukt: "klopt" of "klopt niet". In het voorbeeld klopt het niet: van `0` naar `2` via `1` is samen `3` plus `7`, oftewel `10`, en rechtstreeks is maar liefst `18`.
c. Stel je doet het volgende. Je begint in plaats `i`. Ga nu naar de plaats `i+1` (naar `0` als je in plaats `n-1` zat), enzovoorts. Dit doe je alleen als de lengte van deze stap groter is dan de vorige stap (voor de eerste stap is dit automatisch goed). Schrijf een C++-functie `int ga (int A[n][n], int i)` die uitrekent hoever je dan in totaal bent gekomen. Gebruik een `while`-loop. In het voorbeeld, met `i` gelijk aan `1`: vanuit `1` doe je een stap van `7` naar `2`, vanuit `2` een stap van `18` naar `0`, en daar blijf je: de stap van `0` naar `1` ter grootte `3` is kleiner dan `18`; totaal gelopen: `25`.

Zodra het tentamen is nagekeken, naar alle waarschijnlijkheid volgende week, worden de uitslagen via de webpagina

`http://www.liacs.nl/home/kosters/lst/`

bekend gemaakt. Het tentamen kan dan worden opgehaald in kamer 159 van het Gebouw van Wiskunde en Informatica, Niels Bohrweg 1, Leiden.