Tentamen Inleiding Programmeren voor LS&Ters
Maandag 26 augustus 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 middel (double x, double y, double z)` die het *middelste in grootte* van de drie verschillende getallen `x`, `y` en `z` teruggeeft (met behulp van een `return`-statement). Voorbeeld: bij 3,7,2 is 3 het resultaat.
b. Schrijf een C++-functie `int fac (int n)` die `n!` $= n \times (n-1) \times \cdots \times 3 \times 2 \times 1$ berekent.
c. Schrijf een C++-functie `int binom (int n, int k)` die de binomiaalcoefficient

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

uitrekent. Gebruik **b**.
d. Stel we hebben een array `woordje` met `6` letters (`char woordje[6]`) en een array `verhaal` met `100` letters (`char verhaal[100]`). We willen weten of het `woordje` ergens in het `verhaal` voorkomt: op `6` 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` 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 `void sorteer (double A[n])` die het array `A` oplopend sorteert met *bubblesort*. Gebruik de functie van **a**.
c. Hoeveel vergelijkingen tussen array-elementen doet het algoritme van **b**?
d. Stel dat het array `A` oplopend gesorteerd is. We maken één willekeurig array-element `3.14`, en verstoren daarmee waarschijnlijk de sortering. We willen `A` opnieuw, en met weinig inspanning, sorteren. Geef kort *in woorden* een methode aan die dit doet.

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:

```
void abcde (int & x, int & y, int & z) {
  int temp = x; x = y; y = z; z = temp; x--; y--; z--;
  cout << x << y << z << endl;
}  // abcde
int fghij (int & x, int & y, int & z) {
  if ( z > x ) abcde (y,z,x); else abcde (x,y,z);
  cout << z << x << y << endl; return x + z;
}  // fghij
```

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

```
x = 4; y = 17; z = 25;
abcde (x,y,z); cout << fghij (x,y,z) << endl;
cout << x << y << z << endl;
```

Wat wordt er afgedrukt? Geef hierbij uiteraard uitleg.
c. Als **b**, maar met weglating van de zes `&`'s.
d. Als **b.** (dus met alle zes `&`'s er bij!), voor

```
x = 10; y = 12; z = 15;
abcde (x,x,x); cout << fghij (x,x,x) << endl;
cout << x << y << z << endl;
```

e. Mag ergens in het programma de aanroep `abcde (fghij (x),x,x)` voorkomen? Leg uit.

4. Deze som gaat over een 2-dimensionaal array `int A[n][n];` met `n` rijen en `n` kolommen. Twee voorbeelden met `n = 3`:

```
 1  4  0      2  4  1
16  6  7     16  7  7
 2 24 11      2 24 11
```

a. Schrijf een C++-functie `int aantal (int A[n][n], int X)` die bepaalt hoevaak het getal `X` in het array `A` voorkomt. In het rechter array komt `7` twee keer voor.
b. Schrijf een C++-functie `bool uniek (int A[n][n])` die bepaalt alle getallen uit `A` precies één keer voorkomen; zo ja, dan moet de functie `true` retourneren, en anders `false` Voor de linker matrix uit het voorbeeld moet het `true` zijn, voor de rechter `false`. Gebruik **a**.
c. Schrijf een C++-functie `void doe (int A[n][n], int i, int j)` die de onmiddellijk aangrenzende horizontale en verticale buren van het getal uit de `i`-de rij, `j`-de kolom, met `1` ophoogt. In het voorbeeld ontstaat de rechter uit de linker matrix door de aanroep `doe (A,0,1)`: de drie buren van `4` worden opgehoogd.
d. Schrijf een C++-functie `void verander (int A[n][n])` die één voor één de elementen op de hoofddiagonaal van `A` afloopt, te beginnen links boven (in de linker matrix van het voorbeeld bestaat de hoofddiagonaal uit `1`, `6` en `11`), en telkens de functie `doe` van **c** aanroept. De functie moet stoppen als niet alle elementen van `A` uniek meer zijn (gebruik **b**) of als de hele hoofddiagonaal is behandeld.

Zodra het tentamen is nagekeken, naar alle waarschijnlijkheid eind deze 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.