Tentamen Programmeermethoden
Maandag 2 augustus 2004, 10.00–13.00 uur
Universiteit Leiden — Informatica

Bij alle te schrijven functies moeten de variabelen in de heading of als locale variabele voorkomen (niet stiekem globale variabelen gebruiken). De opgaven tellen alle vier even zwaar mee. Veel succes! Cijfers: `http://www.liacs.nl/home/kosters/pm/res03.txt`.

1. Gegeven een array `A` (`double A[n];`, met `const n = 1000;`) met `n` verschillende getallen.
a. Schrijf een C++-functie `int grootste (double A[ ], int i, int j)` die de *index* oplevert van de grootste waarde uit `A[i]` tot en met `A[j]`. Neem aan dat $0 \leq$ `i` $\leq$ `j` $<$ `n`.
b. Geef een C++-functie `double eenna (double A[ ], int n)` die 't op-een-na-grootste array-element retourneert. Dit resultaat moet worden gegenereerd door de functie van **a** *twee of drie keer* aan te roepen, en de resultaten handig te combineren.
c. Schrijf een C++-functie `void wissel (double A[ ], int i, int j)` die `i`-de en `j`-de array-element (dus `A[i]` en `A[j]`) verwisselt.
d. Schrijf nu een C++-functie `void sorteer (double A[ ], int n)` die 't array oplopend sorteert door herhaald `grootste` en `wissel` aan te roepen met geschikte parameters.
e. Is deze methode beter of slechter dan bubblesort, of wellicht daarmee vergelijkbaar?

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 duidelijk uit.
b. Een zeker C++-programma bevat de volgende programmaregels:

```
int wisselsom (int & a, int & b) {
  int hulp = a; a = b; b = hulp; cout << a << b << hulp << endl;
  return a + b;  }//wisselsom
int xkeer (int & a, int & b, int & x) {
  int aantal = 0;
  while ( x > 0 ) { aantal += wisselsom (a,b); x--;  }//while
  cout << a << b << x << aantal << endl; return aantal;  }//xkeer
```

Gegeven de volgende programma-code:

```
a = 5; c = 3; aantal = 5;
cout << xkeer (a, c, aantal); cout << a << c << aantal << endl;
```

Wat wordt er afgedrukt? Geef hierbij uiteraard uitleg. De variabelen `a`, `c` en `aantal` zijn globaal en van type `int`.
c. Als **b**, maar nu met alle vijf `&`'s weggelaten.
d. Weer terug naar de situatie van **b**, met de vijf `&`'s erbij. Wat gebeurt er bij uitvoering van het volgende stukje C++? Wat wordt er afgedrukt?

```
a = 6; cout << xkeer (a, a, a) << endl; cout << a << endl;
```

e. En wat levert `xkeer (a, b, x)` in het algemeen op als returnwaarde — als functie van `a`, `b` en `x`? Dit voor de situatie van **c**, dus met alle vijf `&`'s weggelaten.

f. Mag een statement als `a = xkeer (wisselsom (a, c), wisselsom (c, a), a);` ergens in het programma staan? En zo ja, is er dan sprake van recursie?

3. We hebben een 2-dimensionaal array `Hoogte` met `m` rijen en `n` kolommen, gevuld met positieve gehele getallen, die de hoogte in een landschap aangeven: `int Hoogte[m][n];`, bijvoorbeeld:

```
 8  8  8  5
 1  2  5  5
 7  7 11  2
```

a. Geef een C++-functie `int vlakte (Hoogte)` die oplevert hoe lang de langste horizontale serie aaneengesloten array-elementen met dezelfde hoogte is (3 in het voorbeeld, wegens `8-8-8`). Vul zelf de heading goed in!
b. Geef een C++-functie `int verschil (Hoogte, i, j1, j2)` die bepaalt wat het totaal overbrugde hoogteverschil is als je horizontaal van `(i,j1)` naar `(i,j2)` gaat. Neem aan dat `j1` $\leq$ `j2`. Voor `i = 2`, `j1 = 0` en `j2 = 3` geeft dit $(7-7)+(11-7)+(11-2) = 13$.
c. Idem, maar met de volgende aanpassing. We mogen één keer naar een eerdere rij (als die er is); we mogen heen en terug op zelfgekozen punten. De functie moet nu de *laagst mogelijke* waarde teruggeven. In het voorbeeld van **b** is het antwoord 11 wegens de route `7-7-2-5-5-2`. Gebruik de functie `verschil`.

4. Gegeven twee enkelverbonden niet-lege lijsten van vakken, die samen alle colleges bevatten die in een zeker studiejaar aan de universiteit gegeven worden. De lijsten bevatten de vakken uit het najaarssemester resp. het voorjaarssemester en zijn toegankelijk via een pointer `najaar` resp. `voorjaar` van type `vak`*.

```
class vak { public: // een struct mag ook
  int code, stp;    // code en stp voor het college
  char docent;      // unieke code voor de docent
  vak* volgende;    // pointer naar het volgende college
};  // vak
```

a. In het curriculum worden twee vakken van semester verwisseld. Dit zijn toevallig het eerste college uit de najaarslijst en het eerste college uit de voorjaarslijst. Schrijf een C++-functie `verwissel (najaar, voorjaar)` die de twee voorste colleges uit de lijsten verwisselt door hun *inhoud* te verwisselen.
b. Door omstandigheden worden de eerste twee colleges uit de najaarslijst dit jaar niet gegeven. Schrijf een C++-functie `gooiweg (najaar)` die deze colleges verwijdert en vernietigt — mits ze bestaan.
c. Schrijf een C++-functie `erbij (najaar, c, s, d)` die een nieuw vak vooraan de najaarslijst toevoegt, met inhoud `c`, `s`, `d`.
d. We willen nu van de twee lijsten een grote lijst maken die alle colleges van het studiejaar bevat. We doen dit door de voorjaarslijst in zijn geheel achter de najaarslijst te hangen. Schrijf een C++-functie `concateneer (colleges, najaar, voorjaar)` die aldus een lijst van alle colleges oplevert. De pointer `colleges` van type `vak*` wordt de ingang van de nieuwe lijst (bij aanroep is deze `NULL`). Na afloop zijn zowel `najaar` als `voorjaar` `NULL`.
e. In de functies bij **a**, **b**, **c** en **d** staat in de heading de parameter `najaar` en/of `voorjaar`. 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.