Tentamen Programmeermethoden
Vrijdag 19 maart 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` (`const n = 1000; double A[n];`) met `n` verschillende getallen.
a. Schrijf een C++-functie `int kleinste (A, i, n)` die de *index* oplevert van de kleinste waarde uit `A[i]` tot en met `A[n-1]`. Vul zelf de heading goed in!
b. Geef een C++-functie `void schuif (A, i, j)` die de elementen `A[i]` tot en met `A[j-1]` alle één plaats naar rechts opschuift en `A[j]` op positie `i` zet. Neem aan dat `i` $<$ `j` $<$ `n`. Voorbeeld: `A` is 1.0 2.0 6.0 8.0 5.0 3.0 4.0; `i` $= 2$; `j` $= 5$; dan is het gevolg van `schuif`: 1.0 2.0 3.0 6.0 8.0 5.0 4.0 (hier is `n` gelijk aan 7).
c. Schrijf nu een C++-functie `sorteer (A, n)` die het array oplopend sorteert door herhaald `kleinste` en `schuif` aan te roepen. Doe dit net als bij "simpelsort": het beginstuk is steeds oplopend gesorteerd, zoek herhaald de kleinste van de (steeds kleiner wordende) rest en schuif die naar het begin van de rest.
d. Stel dat je deze sorteermethode toepast op het rijtje `n n-1 ... 3 2 1`. Hoeveel "verschuivingen" (toekenningen van `double`'s) doet het algoritme van **c** dan in totaal?
e. Idem, nu voor de situatie `0 2 4 6 ... 2(i-1) k 2(i+1) ... 2(n-1)` waarbij we het getal `2i` door een *oneven* getal `k` > 0 kleiner dan `2(i-1)` hebben vervangen.

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 desom (int & een, int & twee) {
  een--; twee++; a--; cout << een << twee << endl;
  return een + twee;  }//desom
double gem (int & a, int & b, int & c) {  // soort gemiddelde
  int deelsom = desom (a, b) + desom (b, c);
  return desom (deelsom, deelsom) / 8.0;  }//gem
```

Gegeven de volgende programma-code:

```
a = 5; c = 4; h = 7;
cout << gem (a, c, h); cout << a << c << h << endl;
```

Wat wordt er afgedrukt? Geef hierbij uiteraard uitleg. De variabelen `a`, `c` en `h` zijn globaal en van type `int`. Zijn er verschillende antwoorden mogelijk?
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 = 7; cout << gem (a, a, a) << endl; cout << a << endl;
```

e. En wat levert `gem (a, a, a)` in het algemeen op — als functie van `a`?
f. Mag een statement als `a = (int) gem (desom (a, c), desom (c, h), h);` ergens in het programma staan? En zo ja, is er dan sprake van recursie?

3. We hebben een 2-dimensionaal array `Life` met `m` rijen en `n` kolommen, gevuld met positieve gehele getallen: `int Life[m][n];` , bijvoorbeeld:

```
3  5  7 ...
1  2  5 ...                hier geeft het getal 3 aan dat er op
7  7 11 ...                positie (0,0) drie levende cellen zitten
...
```

a. Geef een C++-functie `int groter (Life, X, m)` die oplevert hoeveel getallen uit `Life` echt groter zijn dan de integer `X`. Vul zelf de heading goed in!
b. Schrijf een C++-functie `bool max (Life, i, j, m)` die `true` oplevert als `Life[i][j]` het grootste array-element is, en anders `false`. Als het getal meerdere keren voorkomt, en dus niet het enige maximum kan zijn, moet de functie ook `false` geven. Hierbij *moet* de functie van **a** (verstandig) gebruikt worden.
c. Schrijf een C++-functie `void overleef (Life, m)` die alleen die cellen laat overleven die in vakjes zitten waarvoor in totaal tussen 4 en 8 cellen in de directe buren aanwezig zijn. Een en ander gebeurt voor alle array-elementen tegelijk (parallel). Tel voor alle array-elementen de aantallen cellen uit de vier (aan de randen van het array twee of drie) horizontale en verticale buren op, en als dit aantal kleiner dan 4 of groter dan 8 is maak je te zijner tijd het oorspronkelijke array-element 0; anders blijft het onveranderd. In het voorbeeld-array blijft de 3 bestaan, de 5-en wordt 0-en, etcetera.
d. Schrijf een C++-functie `int hoeveel (Life, i, j, m)` die het volgende doet. Neem aan dat `Life[i][j]` niet 0 is. Vanuit `(i,j)` ga je net zolang naar rechts tot je een 0 ziet of de rand, dan naar beneden (in de kolom van de laatste niet-0) tot je weer een 0 of de rand ziet. Daarna weer naar rechts, etcetera, tot je geen van beide kanten meer op kunt. De functie moet de som van alle getallen die je onderweg gezien hebt teruggeven.

4. Een serie dagen is een enkelverbonden lijst van dagen, toegankelijk via de pointer `serie` van type `dag*`.

```
class dag { // een struct mag ook
public: int nummer;     // het nummer van de dag
        dag* eenander;  // wijst naar een of andere dag (of NULL)
        dag* volgende;  // pointer naar volgende dag
};  // dag
```

a. Schrijf een C++-functie `void toevoegen (serie, numero)`, die een nieuwe dag met nummer `numero` vooraan de serie toevoegt. De `eenander`-pointer moet `NULL` worden.
b. Schrijf een C++-functie `void verwijder (serie)`, die de eerste dag uit de lijst verwijdert. Je mag aannemen dat de lijst ten minste twee dagen bevat. Het verwijderen moet gebeuren door de `eenander`-pointer van de nieuwe eerste (de oude tweede) dag naar de oude eerste dag te laten wijzen. Als daar al naar een dag, zeg (×), verwezen werd, moet die echt vernietigd worden. Neem aan dat (×) niet een dag in de lijst zelf betreft.
c. Schrijf een C++-functie `void wissel (serie)`, die de inhoud van de nummervelden van de twee eerste dagen (als die er zijn) in stijgende volgorde zet — door die inhouden eventueel te verwisselen.
d. In de functies bij **a**, **b** en **c** staat in de heading de parameter `serie`. 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 functies.
e. Schrijf een C++-functie `int buiten (serie)`, die het aantal dagen in de lijst bepaalt waarvoor de `eenander`-pointer naar een dag *buiten* de lijst wijst.