Tentamen Programmeermethoden
Maandag 1 augustus 2005, 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/res04.txt`.

1. Gegeven een array `A` (`int A[n];` met `const n = 1000;`) dat `n` verschillende gehele getallen bevat. Merk op dat `n` even is.
a. Schrijf een C++-functie `void verwissel (int A[ ], int i, int j)` die het `i`-de en `j`-de array-element van `A` verwisselt.
b. Schrijf een C++-functie `void reorganiseer (int A[ ], int n)` die alle array-elementen twee aan twee vergelijkt en steeds de grootste van de twee op de even positie zet en de kleinste op de oneven positie. Gebruik **a** indien nodig.
Voorbeeld: `34 76 29 33 88 55 23 47` wordt: `76 34 33 29 88 55 47 23`.
c. Veronderstel vanaf nu dat het array zo is gereorganiseerd als in **b** beschreven. Schrijf nu een C++-functie `int grootste (int A[ ], int n)` die de index van de grootste waarde uit het array retourneert en de grootste waarde zelf afdrukt. Je moet hiervoor alleen de even posities van het array aflopen.
d. We willen nu de op een na grootste waarde uit het array weten. Schrijf daartoe een C++-functie `void eenna (A, n, gr, enagr)` die de index van de op een na grootste waarde via de parameter `enagr` oplevert. Hierin is `gr`, de index van de grootste waarde uit `A`, gegeven. Zorg zelf voor de juiste heading.
e. Hoeveel vergelijkingen (waarbij minstens één array-element betrokken is) kost het in totaal om de op een na grootste waarde uit het array te vinden door achtereenvolgens te reorganiseren (**b**), de grootste (**c**) en de op een na grootste (**d**) te bepalen?

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 onderscheiden we ook nog *formele* en *actuele* parameters. Leg deze zes begrippen duidelijk uit.
b. Gegeven een C++-progamma met daarin de volgende twee functies:

```
int romulus (int p, int q) {
  int i = 6; int x = 0;
  for ( i = 1; i <= p; i++ ) { p--; q--; x += q; }//for
  cout << "romulus: " << x << "," << p << "," << q << "," << i << endl;
  return (x+p+q+i);
}//romulus
void remus (int p, int q) {
  int y = 2; x /= 2;
  p = p + romulus (q,q); q = romulus (y,q) + q + romulus (x,p);
  cout << "remus: " << x << "," << y << "," << p << "," << q << endl;
}//remus
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (alle van type `int`). Voordat de functie `remus` wordt aangeroepen hebben zij de waarde 5, 6 en 4 respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
remus (x,z); cout << "main: " << x << "," << y << "," << z << endl;
```

c. Dezelfde vraag als **b**, maar nu staat er een `&` bij de parameter `p` in `romulus` en bij de parameter `q` in `remus`.
d. Stel nu dat er bij *elke* parameter een `&` staat (bij alle vier dus). Waarom zijn verschillende uitvoeren mogelijk?
e. Je wilt nu binnen de functie `romulus` de functie `remus` aanroepen. Dit is dan (indirecte) recursie. Wat moet je dan nog toevoegen in je programma en waarop moet je letten als het gaat om de werking van het programma?

3. We hebben twee 2-dimensionale arrays `Een` en `Twee` met `m` rijen en `n` kolommen, gevuld met positieve gehele getallen: `int Een[m][n]; int Twee[m][n];` , bijvoorbeeld, met `m = 2` en `n = 3`:

```
3  5  9       3  6  7
2  2  3       1  1  4
```

a. Geef een C++-functie `int aantal (Een, X, tol, m)` die oplevert hoeveel getallen uit `Een` maximaal `tol` (geheel getal ≥ 0) van de integer `X` verschillen. Vul de heading goed in!
b. Schrijf een C++-functie `bool bijnagelijk (Een, Twee, tol, m)` die `true` oplevert als alle overeenkomstige array-elementen uit de twee arrays maximaal `tol` van elkaar verschillen. Voor het voorbeeld `true` als `tol = 3` en `false` als `tol = 1`.
c. Schrijf een C++-functie `int herhaal (Een, Twee, m)` die het volgende doet. Begin in $(0, 0)$ (linksboven). Als de twee array-elementen gelijk zijn, ga één naar rechts, en één naar beneden; als het array-element uit `Een` kleiner is dan dat uit `Twee`, ga één naar beneden, en anders één naar rechts. De functie retourneert het nummer van de stap waarmee je de array's zou verlaten. In het voorbeeld: $(0, 0) \rightarrow (1, 1) \rightarrow (1, 2) \rightarrow (2, 2)$, antwoord 3.
d. Schrijf een C++-functie `int kwa (Een, Twee, m)` die het aantal paren array-elementen (het ene getal uit `Een`, het andere uit `Twee`) uit de twee array's oplevert, waarbij de een het kwadraat is van de ander. In het voorbeeld 3 stuks, namelijk $9 = 3^2$, $2^2 = 4$ en $2^2 = 4$.

4. We hebben twee lijstjes mensen (mannen en vrouwen), die bestaan uit

```
class mens {
public: int leeftijd;    // de leeftijd
        mens* volgende;  // wijst naar volgende mens (of NULL)
};//mens
```

a. Schrijf een C++-functie `void erbij (mannen, vrouwen, jaar, mv)`, die een nieuwe mens met leeftijd `jaar` vooraan een van de twee lijsten `mannen` of `vrouwen` toevoegt. Als de Boolese variable `mv` `true` is wordt aan de vrouwenlijst toegevoegd, anders aan de mannenlijst.
b. Schrijf een C++-functie `void verwijder (vrouwen, mannen)`, die uit beide lijsten de eerste mens verwijdert, mits die bestaat. Als een lijst leeg is moet er voor die lijst niets worden gedaan.
c. Schrijf een C++-functie `void wissel (mannen, vrouwen)`, die de leeftijden van de eerste vrouw en de eerste man verwisselt — mits deze beide bestaan.
d. In de functies bij **a**, **b** en **c** staat in de heading de parameters `mannen` en `vrouwen`. 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 `bool evenoud (vrouwen, mannen)`, die `true` oplevert precies dan als de eerste man en vrouw even oud zijn, de tweede man en vrouw idem, etcetera. Kortom: de lijstjes bevatten precies dezelfde (*en* evenveel) getallen, in dezelfde volgorde.