Tentamen Programmeermethoden
Maandag 9 januari 2006, 14.00–17.00 uur
Universiteit Leiden — Informatica

Bij alle te schrijven functies moeten de variabelen in de heading of als locale variabele voorkomen; vul zelf de headings goed in. De opgaven tellen alle vier even zwaar mee. Veel succes! Cijfers: `http://www.liacs.nl/home/kosters/pm/res05.txt`.

1. We hebben een array `B` met verschillende double's, zoals: `2.0 0.6 3.1 7.2 6.2 6.5`.
a. Schrijf een C++-functie `void gk (B,gr,kl,n)` die in `gr` en `kl` de *array-indices* van het grootste en kleinste getal van `B` (met `n` ≥ 2 elementen) oplevert. Er moet precies één for-loop gebruikt worden. (In het voorbeeld: 3 en 1.)
b. Schrijf een C++-functie `int stijgdaal (B,n)` die bepaalt hoeveel *locale extremen* `B` heeft: array-elementen waarvoor beide directe buren kleiner zijn, of juist beide groter. Eerste en laatste element zijn per definitie ook locale extremen. (Voorbeeld: 5; `3.1` niet.)
c. Schrijf een C++-functie `bool opso (B,n)` die bepaalt of `B` oplopend gesorteerd is. Het array mag alleen met de functies van **a** en **b** benaderd worden. (Voorbeeld: `false`.)
d. Schrijf een C++-functie `int lang (B,n)` die bepaalt wat de lengte is van de langste stijgende serie aaneengesloten array-elementen. (Voorbeeld: 3, namelijk `0.6 3.1 7.2`.)
e. Als **c**, maar gebruik nu alleen de functie van **d**.

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++-programma met daarin de volgende twee functies:

```
bool rita (int m, int n) {
  if ( m < n ) n--; else m--;
  x++; cout << x << ", " << m << " en " << n << endl;
  return ( m < n );
}//rita
int salomon (int x, int y) {
  if ( rita (x,y) ) { x = x - y; y = x + y; x = y - x; }
  else { x = x + 10; y = y - 2; }
  cout << x << " en " << y << endl; return x+y;
}//salomon
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (alle van type `int`). Voordat de functie `salomon` wordt aangeroepen hebben zij de waarde 77, 3 en 4 respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x = salomon (y,z); cout << x << ", " << y << " en " << z << endl;
```

c. Als **b**, maar nu staat er een `&` bij alle vier parameters, en `x`, `y` en `z` starten op 77, 3 en 8.
d. Als **c** (dus met vier `&`'s), en `x`, `y` en `z` beginnen op 995, 3 en 8, voor

```
x = salomon (x,x); cout << x << ", " << y << " en " << z << endl;
```

e. Je wilt nu binnen de functie `rita` de functie `salomon` 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. Gegeven is een `m` bij `n` (beide constanten) array `numbers`, gevuld met gehele getallen.
a. Schrijf een C++-functie `void positief (numbers)` die alle negatieve getallen uit het array positief maakt door het teken om te klappen. Dus bijvoorbeeld: -2006 wordt 2006.
b. We willen weten of er een kolom is waarvan de som der elementen precies gelijk is aan een gegeven waarde `getal`. Schrijf daartoe een C++-functie `int kolom (numbers, getal)` die het kolomnummer oplevert indien zo'n kolom bestaat, en anders -1. Er dient gestopt te worden zodra zo'n kolom gevonden is.
c. Schrijf nu een C++-functie `int achterelkaar (numbers, waarde)` die bepaalt in hoeveel rijen een gegeven geheel getal `waarde` minstens twee keer naast elkaar voorkomt. Er dient per rij steeds gestopt te worden zodra `waarde` twee keer naast elkaar is aangetroffen.
d. Loop als volgt door het array, beginnend in `numbers[0][0]`. Kijk eerst of direct rechts van en direct onder de huidige plek twee dezelfde getallen staan. Zo ja, loop naar de buurplek diagonaal rechts onder. Zo nee, kijk of direct links van en direct onder de huidige plek dezelfde getallen staan. Zo ja, ga naar de buurplek diagonaal links onder, zo nee stoppen. Let erop dat je niet buiten de arraygrenzen komt. Schrijf een C++-functie `void lopen (numbers, rij, kolom)` die zo door het array loopt en het rij- en kolomnummer oplevert waar de wandeling stopt. Gebruik een while-loop.

1	2	7	4
2	4	3	19
10	3	6	18
9	17	3	6
5	11	7	14

Voorbeeld: `m = 5` en `n = 4`; wandeling stopt in rij 3 en kolom 1.

4. Gegeven is het volgende type:

```
class informatie { public:
  informatie* volgende;  informatie* vervolgende;
  int jaartal;                                    };//informatie
```

Met behulp hiervan worden rijtjes (lijstjes) jaartallen opgebouwd. Het veld `volgende` bevat steeds een pointer naar het direct "rechts" naast gelegen vakje, het veld `vervolgende` bevat een pointer naar het daar weer "rechts" naast gelegen vakje — als dat bestaat, anders `NULL`. Een voorbeeld (`ingang` van type `informatie*`):

```
ingang ---> 2006 ---> 1905 ---> 1848 NULL
```

Het vakje met `2006` erin heeft hier nog een pointer `vervolgende` naar het vakje met `1848` erin; voor de vakjes met `1905` en `1848` is deze pointer `NULL`.
a. Schrijf een C++-functie `voegtoe (ingang,nieuw)` die een nieuw vakje met jaartal `nieuw` erin vooraan de structuur (met `ingang` van type `informatie*` als ingang) toevoegt. Denk aan de lege lijst.
b. Schrijf een C++-functie `verwijder (ingang)` die het eerste vakje uit de lijst (met `ingang` van type `informatie*` als ingang) verwijdert indien in dat vakje *niet* het jaartal `2006` zit. Denk aan de lege lijst.
c. Schrijf een C++-functie `verwissel (ingang)` die de inhoud van de `jaartal`-velden van eerste en derde vakje verwisselt — indien deze vakjes bestaan, en anders niets doet.
d. In de functies bij **a**, **b** en **c** staat in de heading de parameter `ingang`. 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 mogelijkheden precies gebeurt tijdens executie van de betreffende functie.
e. Schrijf een C++-functie `int lengte (ingang)` die voor het algemene geval uitrekent wat het aantal vakjes is in de lijst met `ingang` als ingang; voor de voorbeeldstructuur is dat `3`. Voor de lege lijst moet de functie `0` opleveren. Er moet zoveel mogelijk van de `vervolgende`-velden gebruik gemaakt worden.