Tentamen Programmeermethoden
Maandag 4 augustus 2003, 10.00–13.00 uur
Universiteit Leiden — Informatica

Bij alle te schrijven functies moeten de variabelen in de heading voorkomen (niet stiekem globale variabelen gebruiken). De opgaven tellen alle vier even zwaar mee. Veel succes!

1. We hebben een array `A` (`double A[n]`, met `const int n = 12345;`) met `n` verschillende getallen.
a. Geef een C++-functie `void wissel (A,i,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 bubblesort (A,n)` die het array `A` oplopend sorteert met *bubblesort*. De eenvoudigste versie is goed.
c. Geef een C++-functie `double kde (A,k)` die het `k`-de element in grootte (met $1 \leq$ `k` $\leq$ `n`) van het array `A` oplevert. Dat is array-element `A[n-k]` *als* het array gesorteerd zou zijn. Pas daarom de functie van **b** zodanig aan dat na de `k`-de doorgang het sorteren afbreekt, en het antwoord gegeven wordt.
d. Hoeveel vergelijkingen tussen `double`'s doet het algoritme van **c** voor algemene `n` en `k`? (In het antwoord mag een sommatie blijven staan.)
e. Als **c**, maar nu voor een situatie waarin het array `A` gesorteerd is, waarna twee willekeurige elementen zijn verwisseld. Er mogen maximaal `n-1` vergelijkingen tussen `double`'s gedaan worden.

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 hoevaak (int x) {
  g++; x = x + g; cout << g << x << endl; return x;
}  // hoevaak
int doewat (int x, int y) {
  int ik = hoevaak (x), i;
  for ( i = 1; i <= ik; i++ ) y += hoevaak (y);
  cout << g << x << y << ik << i << endl;
  return y;
}  // doewat
```

Wat levert op (met globale variabelen `g`, `a` en `b` van type `int`):

```
g = 0; a = 3; b = 4;
cout << doewat (a,b) << endl; cout << g << a << b << endl;
```

Wat wordt er afgedrukt? Geef hierbij uiteraard uitleg.
c. Als **b**, maar met drie `&`'s in de headings van `doewat` en `hoevaak`.
d. Idem, als bij **c**, dus met de drie `&`'s erbij, maar nu voor

```
g = 0; a = 3; b = 4;
cout << doewat (g,g) << endl; cout << g << a << b << endl;
```

e. Stel dat je in de functie `hoevaak` de functie `doewat` zou willen aanroepen. Mag dat, en zo nee, wat moet je doen om het wel te mogen laten kunnen? En is er dan sprake van recursie?

3.a. Schrijf een functie `void initaliseer (...)` die als parameter een matrix van $M \times N$ integers meekrijgt en hiervan alle elementen op 0 initialiseert, behalve element '$0, 0$' dat de waarde 1 krijgt.
b. Schrijf een functie `volgende (...)` die, gegeven een integer array `route` van lengte n (waarbij n even) en een integer `index` (eveneens even) de waarde teruggeeft van de volgende twee elementen van het array (het `index`-de en het element ernaast; gebruik hiervoor call-by-reference variabelen). Indien deze niet bestaan, wordt $M \times N$ teruggegeven in beide variabelen.

Op veld $(0, 0)$ van een bord (een matrix) van $M \times N$ staat een speler met een vreemd soort routebeschrijving: een die bestaat uit een array van integers. Elk oneven array-element geeft aan wat de horizontale verplaatsing moet zijn in de volgende stap en elk even element geeft aan wat de verticale verplaatsing moet zijn.
c. Schrijf een functie `stap (...)` die gegeven een matrix van $M \times N$, een huidige positie 'x, y' en een gewenste verplaatsing 'dx, dy' het getal 1 zet op de "nieuwe" plaats bepaald als '$x + dx, y + dy$'. Als een stap "onmogelijk" is (bijvoorbeeld omdat een coordinaat negatief dreigt te worden), dan dient een foutmelding te worden gegeven en de waarde -1 te worden teruggegeven. Anders wordt de *oorspronkelijke* waarde van de nieuwe plaats teruggegeven.
d. Gegeven de functies in de vorige onderdelen, schrijf een functie `spring (...)` die vanaf positie $0, 0$ in de matrix alle sprongen uitvoert die in het array `route` staan. Hierbij gelden de volgende regels:

- Als een stap niet kan dan dient het programma te worden beeindigd met een foutmelding.
- Als een positie meer dan eens wordt aangedaan dienen de coordinaten van de positie te worden afgedrukt met daarbij de mededeling dat deze positie al eens bereikt is (het programma gaat wel door).
- Als de gehele route "afgelopen" is dient te worden afgedrukt dat het einddoel is bereikt (met daarbij de eindpositie).

4. Gegeven is een klasse rits als volgt:

```
class rits {
  public:
    vakje *lijst1;
    vakje *lijst2;
    void lijst2voegtoe (int waarde);
    void ritsdicht ( );
};
```

Een `vakje` bestaat uit een integer (`info`) en een pointer (`volgende`) naar een volgend vakje. De methode `lijst2voegtoe (...)` neemt de integer parameter, maakt een vakje aan met die waarde, en voegt dit vakje *vooraan* `lijst2` toe.
a. Geef de code voor methode `lijst2voegtoe (...)`. Hou in dit onderdeel en alle volgende onderdelen rekening met het feit dat een lijst `NULL` kan zijn.
b. Geef de code voor een functie `lijst1vernietigvakje (rits &r)` die het voorste vakje van `lijst1` verwijdert en vernietigt.
c. Geef de code voor een functie `int lijst2lengte (rits &r)` die de lengte van `lijst2` teruggeeft.
d. Geef de code voor een functie `void lijst1VoegIn (rits &r, vakje &v)` die een vakje `v` invoegt na het eerste element van `lijst1` (als dat bestaat, anders wordt `v` het eerste vakje).

De methode `ritsdicht ( )` "ritst" de twee lijsten in elkaar. Dat wil zeggen: hij verwijdert het voorste `vakje` van `lijst2` en voegt dit toe direct achter het voorste element van `lijst1`. Vervolgens verwijdert hij het tweede element van `lijst2` en voegt dat toe achter het volgende element van `lijst1`. Als `lijst2` langer is dan `lijst1` dan worden de overgebleven elementen van `lijst2` er achteraan geplakt, zoals in de schets hieronder.

```
lijst1: 1-2-3-4      'ritsdicht'  nieuwe lijst1: 1-5-2-6-3-7-4-8-9-10
                     ---------->
lijst2: 5-6-7-8-9-10              nieuwe lijst2:
```

e. Geef de code voor de methode `ritsdicht ( )`.