Tentamen Programmeermethoden
Donderdag 28 maart 2013, 10:00–13:00 uur
Universiteit Leiden — Informatica

Bij alle functies moeten de variabelen (constanten eventueel uitgezonderd) in de heading of als locale variabele voorkomen; vul zelf headings goed in. De opgaven tellen alle vier even zwaar mee. Succes! Cijfers: `http://www.liacs.nl/home/kosters/pm/cijf/res.html`.

1. In een array `int A[n]` staan `n` (een `const` > 0) gehele getallen.
a. Schrijf een Booleaanse C++-functie `uniek (A,i,n)` die precies dan `true` teruggeeft als het `i`-de array-element precies één keer in `A` voorkomt, en anders `false`. Neem aan dat $0 \leq$ `i` $<$ `n`. Als `A` is 7 3 4 7 2: met `i` $= 2$ `true` (getal 4), met `i` $= 3$ `false` (getal 7).
NB Geef hier (en ook in de andere opgaven) de compleet ingevulde heading van de functie!
b. Schrijf een Booleaanse C++-functie `som (A,X,i,j,n)` die precies dan `true` teruggeeft als er twee array-elementen in `A` zijn met als som de integer `X`, en anders `false`. De parameters `i` en `j` (met `i` $<$ `j`) moeten indices worden van dergelijke array-elementen (`-1` als deze niet bestaan). Als er keuze is: het eerste tweetal dat je vindt.
c. Schrijf een C++-functie `sorteer (A,n)` die het array `A` *aflopend* sorteert met behulp van *bubblesort*. Geef een versie die altijd precies `n`$(n-1)/2$ vergelijkingen tussen array-elementen doet.
d. Schrijf de functie van **a** opnieuw, maar nu onder de aanname dat `A` aflopend gesorteerd is. Er mogen maximaal drie array-elementen bekeken 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 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:

```
int ben (int x, int y) {
  x++; y--; z++; cout << "B" << x << "," << y << "," << z << endl;
  return x+y+z;
}//ben
int frans (int x, int y) {
  int z = ben (ben (x,y),y); cout << "F" << x << "," << y << "," << z << endl;
  return z;
}//frans
```

Verder zijn de globale variabelen `x`, `y` en `z` van type `int` gegeven. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x = 1; y = 4; z = 6; cout << frans (x,y) << endl;
cout << x << "," << y << "," << z;
```

c. Als **b**, maar nu met `&`'s voor beide `y`'s in de headings van `ben` en `frans`. Geef aan wat het verschil is tussen de situaties waarin bij `ben (ben (x,y),y)` de door de binnenste aanroep `ben (x,y)` gewijzigde `y`, respectievelijk de oorspronkelijke `y` voor de tweede parameter gebruikt wordt.
d. Stel dat voor alle vier voorkomende parameters in de functie-headings een `&` wordt gezet. Wat gebeurt er nu, en waarom?
e. Wat is *recursie*, en is daar sprake van bij het gedeelte "`int z = ben (ben (x,y),y);`"?

3. Gegeven is een `m` bij `n` (beide `const` > 0) array `G`, gevuld met integers tussen 0 en 9. De rijen stellen getallen tussen `1` en $10^{\texttt{n}} - 1$ voor. Zo representeert rij 1 in het voorbeeld het getal 178.

0	3	9	8	0
0	0	1	7	8
2	1	2	5	0

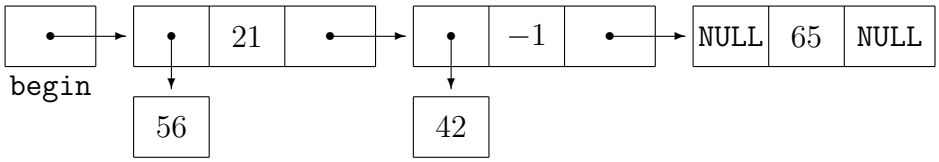
a. Schrijf een C++-functie `int rij (G,i)` die berekent welk getal de `i`-de rij uit `G` voorstelt. Neem aan dat $0 \leq \texttt{i} < \texttt{m}$, en dat het betreffende getal maximaal `INT_MAX` is.
b. Schrijf een C++-functie `telop (G,i,j,k)` die de `i`-de rij en de `j`-de rij van `G` optelt naar de `k`-de rij. Neem aan dat $0 \leq \texttt{i} < \texttt{j} < \texttt{k} < \texttt{m}$, en dat de som kleiner is dan $10^{\texttt{n}}$ en `INT_MAX`. Tip: gebruik de functie van **a**, maar het hoeft niet.
c. Schrijf een C++-functie `int groot (G)` die de array-index oplevert van de rij van `G` die het grootste getal voorstelt. Als dit maximum door meerdere rijen wordt gerealiseerd, geef dan de grootste index die dat doet. In het voorbeeld: 2. Gebruik de functie van **a**.
d. Besloten wordt getallen "links aan te schuiven", waarbij de eventuele nullen die dan rechts ontstaan vervangen worden door het getal `10`. Schrijf een C++-functie `doe (G)` die `G` zo wijzigt.

3	9	8	0	10
1	7	8	10	10
2	1	2	5	0

4. Gegeven is het volgende type:

```
class persoon { public: int* nummer; int leeftijd; persoon* rechts; };
```

Hiermee wordt een lijst met personen gemaakt. Het veld `rechts` bevat een pointer naar het eerstvolgende `persoon`-object. De `nummer`-pointer bevat een *pointer naar een integer* (`NULL` als het nummer onbekend is), en het veld `leeftijd` bevat de leeftijd (-1 als deze onbekend is). Een voorbeeld, met `begin` van type `persoon*`:



a. Schrijf een C++-functie `voegtoe (begin,nr,lt)` die een nieuw `persoon`-object met de integer `nr` als nummer, en de integer `lt` als leeftijd, vooraan de structuur met ingang `begin` van type `persoon*` toevoegt; hierbij staat de waarde -1 voor "onbekend" (dat wordt dus een `NULL`-pointer in geval van `nr`). Tip: één of twee keer `new`!
b. Schrijf een C++-functie `verwijder (begin)` die het eerste `persoon`-object uit de structuur (met `begin` van type `persoon*` als ingang) netjes verwijdert, mits dat object bestaat. Denk ook aan het goed verwijderen van de eventuele integer!
c. Schrijf een C++-functie `ruilom (begin)` die de nummers van de twee eerste `persoon`-objecten omruilt — mits deze objecten bestaan. Hierbij moeten de *pointers* verwisseld worden, in het voorbeeld die naar 56 en 42.
d. In de functies bij **a**, **b** en **c** staat in de heading een pointer. 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? Mag het, moet het? Leg duidelijk uit.
e. Schrijf een C++-functie `indewar (begin)` die de leeftijden als volgt "in de war" brengt: de leeftijd van ieder `persoon`-object moet de leeftijd van het eerstvolgende `persoon`-object worden waarvan de leeftijd bekend is. Als dit niet bestaat moet de leeftijd onveranderd blijven. In het voorbeeld wordt 21 overschreven door 65, terwijl 65 ongewijzigd blijft (evenals -1).