Tentamen Programmeermethoden
Donderdag 12 maart 2015, 14:00–17:00 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` ≥ 1) gehele getallen > 0.
a. Schrijf een C++-functie `int hoe (A,X,n)` die bepaalt hoe vaak het gehele getal `X` in het array `A` voorkomt.
b. Schrijf een C++-functie `uniek (A,n)` die voor ieder getal dat in `A` voorkomt alleen het eerste optreden laat staan, en alle volgende vervangt door `0`. Zo moet `2 5 3 2 2 4 5 6` worden: `2 5 3 0 0 4 0 6`. (De functie van **a** is hiervoor niet nodig.)
c. Schrijf een C++-functie `busort (A,n)` die het array `A` met behulp van *bubblesort aflopend* sorteert, waarbij het algoritme stopt zodra er in een ronde geen verwisselingen waren.
d. Schrijf opnieuw de functie van **b**, nu onder de aanname dat het array aflopend gesorteerd is. Ieder array-element mag maar één keer bekeken (en zonodig direct gewijzigd) worden.
e. Hoeveel if-statements (uitgedrukt in `n`) worden maximaal uitgevoerd in de functies van **b**, **c** en **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 ivo (double & r) {
  r = r + 1.0; int j = (int) r;  // rondt naar beneden af!
  if ( r - j < 0.01 ) return true; else return false;
}//ivo
void fred (double & r, double & s, double & t) {
  int z = 42; r = r + s; s = r - s; r = r - s;
  if ( ivo (t) ) r += t + 10; else s += t;
  r = r + (int) ivo (t);
  cout << r << s << t << z << endl;
}//fred
```

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

```
r = 1.1; y = 2.2; z = 3.3; fred (y,r,z); cout << r << y << z << endl;
```

c. Idem, maar nu zonder de vier `&`-s in de headings van `ivo` en `fred`.
d. Wat levert op (met de `&`-s er weer bij):

```
r = 3.0001; y = 4.0001; z = 8.0001; fred (r,r,r); cout << r << y << z << endl;
```

Er zijn verschillende uitkomsten mogelijk; geef deze. Geef duidelijke uitleg.
e. Als in de functie `ivo` ergens `fred (r,r+1,r+2);` staat, compileert het programma dan nog? Onderscheid gevallen met en zonder `&`.

3. Gegeven is een `m` bij `n` (beide `const` $>$ `0`) array `V` met gehele getallen. Hierbij geeft `V[i][j]` het aantal verkopen van artikel `j` in winkel `i` aan (met $0 \leq$ `i` $<$ `m` en $0 \leq$ `j` $<$ `n`). Zie hiernaast voor een voorbeeld met 4 winkels en 5 artikelen. De constantes `m` en `n` hoeven bij deze opgave niet doorgegeven te worden als parameter.

3	0	4	2	0
7	2	4	4	2
1	2	0	4	1
6	9	3	1	0

a. Een winkel is *beter* dan een andere winkel, als ieder artikel minstens even vaak verkocht wordt. Schrijf een Booleaanse C++-functie `beter (V,i1,i2)` die bepaalt of winkel `i1` beter is dan winkel `i2`. Neem aan dat $0 \leq$ `i1`, `i2` $<$ `m` en `i1` $\neq$ `i2`. In het voorbeeld geldt dat `beter (V,1,0)` `true` is.

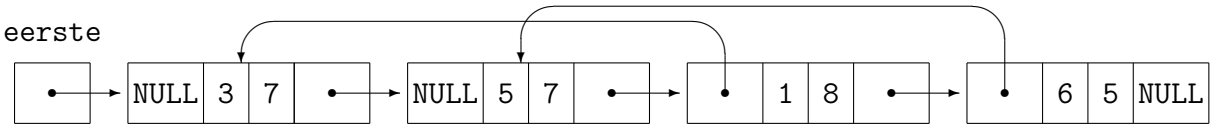
b. Schrijf een C++-functie `int meest (V,tel,aantal)` die de index van *een* winkel met in totaal de meeste verkochte artikelen oplevert. In `tel` moet het aantal van deze winkels worden opgeleverd, in `aantal` het maximale totaal. In het voorbeeld: `1` (of `3`), `19` in `aantal`, voor `tel` $=$ `2` winkels.

c. Een manager bezoekt winkels als volgt. Hij begint bij winkel `i1`, en zoekt daar het eerste product dat niet is verkocht. Daarna gaat hij/zij naar de *eerstvolgende* winkel waar dat betreffende product minstens 4 keer wordt verkocht. Daar zoekt hij/zij het *eerstvolgende* product dat niet is verkocht, enzovoorts. Het proces stopt als het gezochte product of de gezochte winkel niet bestaat. Schrijf een C++-functie `int manage (V,i1)` die uitrekent hoeveel winkels bezocht worden. Neem aan dat $0 \leq$ `i1` $<$ `m`. In het voorbeeld levert `manage (A,0)` `2` op: eerst wordt `0` bezocht, daarna `3`. En `manage (A,1)` geeft `1`.

4. Gegeven is het volgende type:

```
class vak { public: vak* voorvorig; int nr; int cijfer; vak* volg; };
```

Hiermee wordt een dubbel-verbonden lijst van vakken gemaakt. Het veld `volg` bevat een pointer naar het volgende `vak`-object, en `voorvorig` naar het voor-vorige; of `NULL` als dit object niet bestaat. Een voorbeeld (`eerste` van type `vak*`):



a. Schrijf een C++-functie `voegtoe (eerste,vaknr,vakcf)` die een nieuw `vak`-object met `vaknr` en `vakcf` erin vooraan in de lijst met ingang `eerste` toevoegt.

b. Schrijf een C++-functie `verwijder (eerste)` die het voorste `vak`-object uit de structuur die door `eerste` wordt aangewezen, netjes verwijdert — mits het bestaat.

c. Schrijf een C++-functie `wissel (eerste)` die de vak-nummers van de twee voorste vakken omwisselt, mits de cijfers gelijk waren (zoals in het voorbeeld, waar 3 en 5 verwisseld moeten worden). Controleer of de lijst wel minstens twee objecten heeft.

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 `int rangnummer (eerste,ptr)` die bepaalt welk rangnummer het door de pointer `ptr` aangewezen vak heeft: het hoeveelste vak is het? Neem aan dat `ptr` een vak in de niet-lege lijst met ingang `eerste` aanwijst. De functie mag alleen de `voorvorig`-pointers gebruiken, en geen enkele `volg`-pointer. In het voorbeeld: als `ptr` naar het laatste vak (het vierde dus) wijst, levert de functie `4` op.