Tentamen Programmeermethoden
Donderdag 13 maart 2014, 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 `double A[n]` staan `n` (een `const` > 1) verschillende double's.
a. Schrijf een C++-functie `double groot (A,X,n)` die het grootste getal uit het array `A` dat verschilt van de double `X`, teruggeeft (ongeacht of `X` zelf in `A` voorkomt).
b. Schrijf een C++-functie `double grootste (A,n)` die het grootste getal uit het array `A` teruggeeft. Hierbij mag `A` alleen met de functie van **a** benaderd worden, dus ook `A[i]` mag niet rechtstreeks gebruikt worden. Tip: kies handige waarde(s) voor `X`.
c. Tijdens het sorteren met behulp van *bubblesort* staan na de `k`-de ronde de `k` grootste getallen uit `A` achteraan. Schrijf een functie `double kde (A,k,n)` die zo het `k`-de element in grootte bepaalt: na de `k`-de ronde moet gestopt worden, waarna het resultaat wordt geretourneerd. Het array `A` is dan deels gewijzigd, overigens. Neem aan dat $1 \leq$ `k` $\leq$ `n`.
d. Als met de methode van **c** het kleinste getal van `A` wordt uitgerekend, met `k` $=$ `n`, gaat dat niet efficiënt. Hoeveel vergelijkingen tussen double's doet **c** dan, en hoeveel doet een efficiënte methode er?

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 kees (int n, int u, int v) {
  for ( a = 1; a <= n; a++ ) {
     u = u + v; v = u - v + 1; u = u - v + 2;
     cout << a << "," << u << "," << v << endl; }//for
  return u+v;
}//kees
int wim (int x, int y) {
  if ( x > y ) y = kees (x-y,x,y); else x = kees (y-x+1,y,x);
  cout << x << "," << y << endl; return x+y;
}//wim
```

Verder zijn de globale variabelen `a` en `b` gegeven (van type `int`). Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit; wat is de waarde van `a` na de for-loop?):

```
a = 6; b = 2; b = wim (a,b); cout << a << "," << 2 * b - 2 << endl;
```

c. Als **b**, maar nu met een `&` ("ampersand") bij de parameters `u`, `v`, `x` en `y` van de functies.
d. Als **c**, dus met vier `&`'s, maar nu voor:

```
a = 6; b = 2; b = wim (b,b); cout << a << "," << 5 * b + 2 << endl;
```

e. Geef een eenvoudige uitdrukking voor de return-waarde van een aanroep `kees (n,u,v)`, uitgedrukt in `n`, `u` en `v`. Zet hierbij geen `&` bij de parameters.
f. Als in de functie `kees` ergens `u = wim (u,v+1);` staat, *compileert* het programma dan nog? Onderscheid gevallen met en zonder `&`.

3. Gegeven is een `m` bij `n` (beide `const` > 0) array `Tekst` met hoofdletters. Voor ieder array-element geldt dat de letter er onder en er rechts naast (mits deze bestaan) van elkaar verschillen (#). De constantes `m` en `n`, en later ook `w`, hoeven bij deze opgave niet doorgegeven te worden als parameter.

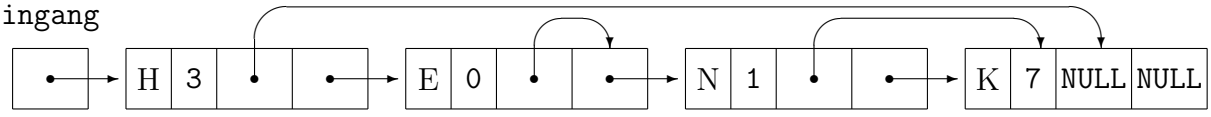
```
B  M* A* R  A  T
T  L  G* E* R* X
U  U  U  U  U  U
```

a. Schrijf een Booleaanse functie `okee (Tekst)` die de eis (#) voor het array `Tekst` controleert. In het voorbeeld is dit `true`.
b. Schrijf een functie `int verschillende (Tekst,i)` die telt hoeveel verschillende letters de `i`-de rij van `Tekst` bevat. In het voorbeeld: `5` voor rij `0`, `6` voor rij `1` en `1` voor rij `2`.
c. Gegeven een array `char Woord[w]` waarin een woord van `w` (een `const` > 0) hoofdletters zit. Een dergelijk woord komt in `Tekst` beginnend op positie $(\texttt{i}, \texttt{j})$ voor als je de letters vanaf daar in volgorde kunt vinden door steeds of naar rechts of naar beneden te gaan. In het voorbeeld, met $\texttt{w} = 5$, komt `M A G E R` voor op positie $(0, 1)$, zie de `*`-en. Schrijf een Booleaanse functie `komtvoor (Tekst,Woord,i,j)` die dit controleert. Neem aan dat $0 \leq \texttt{i} < \texttt{m}$ en $0 \leq \texttt{j} < \texttt{n}$.

4. Gegeven is het volgende type:

```
class letter { public: char naam; int aantal; letter* vervolg; letter* volg; };
```

Hiermee wordt een lijst met `letter`-objecten gemaakt. Het veld `volg` bevat een pointer naar het volgende `letter`-object (of `NULL`), het veld `vervolg` bevat een pointer naar het `aantal` (≥ 0) verderop gelegen `letter`-object, waarbij de pointer `NULL` is als dat object er niet is. Een voorbeeld (`ingang` van type `letter*`):



a. Schrijf een C++-functie `voegtoe (ingang,denaam,getal)` die een nieuw `letter`-object met `denaam` en `getal` erin vooraan de lijst met `ingang` toevoegt. De pointer `ingang` moet naar het nieuwe `letter`-object gaan wijzen. Neem aan dat `getal` `0` of `1` is, en zet de `vervolg`-pointer ook goed.
b. Schrijf een C++-functie `verwijder (ingang)` die het eerste `letter`-object uit de lijst, dat door `ingang` van type `letter*` wordt aangewezen, netjes verwijdert. Controleer of de lijst leeg is.
c. Schrijf een C++-functie `wissel (ingang)` die `naam`- en `aantal`-velden van het eerste en tweede object omwisselt, mits deze bestaan en mits het `aantal` van het tweede object precies 1 meer is dan dat van het eerste. Zet ook de `vervolg`-pointers goed.
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 `zet (ingang,waarde)` die alle `aantal`-velden uit de lijst met `ingang` op `waarde` zet, en de `vervolg`-velden juist aanpast. Neem aan dat `waarde` ≥ 0.