Tentamen Programmeermethoden
Donderdag 14 maart 2019
14:00–17:00 uur Informatica

Universiteit Leiden
The Netherlands

Bij alle functies moeten de variabelen (constanten uitgezonderd) in de heading of locaal voorkomen; vul zelf headings goed in. De te behalen punten (totaal 100) staan tussen haakjes bij de opgaven. Succes! Cijfers: te zijner tijd via Blackboard en uSis.

1. (25) In het array `int A[n]` staan `n` (een `const int` ≥ 2) gehele getallen.
a. (7) Schrijf een C++-functie `klgr (A,start,einde,kl,gr)` die array-indices van kleinste en grootste array-element van `A[start]`, ..., `A[einde]` oplevert in de parameters `kl` en `gr`. Neem aan dat $0 \leq$ `start` $<$ `einde` $<$ `n`. Als er meerdere kandidaten mag je zelf kiezen welke je geeft. (Voorzie alle variabelen in de functie-heading van het juiste type!)
b. (6) Schrijf een Booleaanse C++-functie `palindroom (A,n)` die kijkt of `A` van voor naar achter en van achter naar voor hetzelfde is, zoals bij `7 2 4 4 2 7` en `3 5 1 5 3`.
c. (8) Schrijf een C++-functie `sorteer (A,n)` die het array `A` als volgt oplopend sorteert. Vind met behulp van de functie van **a** kleinste en grootste getal van het nog resterende deel van het array, en wissel dit naar voor en achter, respectievelijk. Doe dit herhaald.
d. (4) Is de sorteermethode van **c** slechter dan, even goed als, of beter dan *bubblesort*? Leg uit; in het antwoord moet `n` voorkomen.

2. (25) **a**. (6) 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. (6) Gegeven een C++-programma met daarin de volgende twee functies:

```
int maart (int p, int q) {
  x++; q++; p--; return ( p + q ); x += 10; }//maart
int april (int x, int y) {
  int s = 3, t = y + 1; s = maart (maart (x,y),t);
  cout << s << "," << t << "," << x << "," << y << endl;
  return ( s + t ); }//april
```

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

```
x = 2; y = 5; cout << april (x,y) << ","; cout << x << "," << y << endl;
```

c. (4) We voegen een `&` toe bij de parameters `q`, `x` en `y` (niet bij `p`). Doe **b** opnieuw.
d. (5) Zet nu alleen twee `&`'s in de heading van `april`, bij `x` en `y`. Wat is de uitvoer van:

```
x = 4; y = 1; cout << april (x,x) << ","; cout << x << "," << y << endl;
```

e. (4) Mag in `maart` ergens `y = maart (april (maart (p,y),y),42);` staan? Onderscheid situaties met/zonder `&`.

3. (25) Gegeven bij deze opgave is een 2-dimensionaal array `int P[m][n];` met zekere `const int m` ≥ 2 en `const int n` ≥ 2. Hierbij staat `P[i][j]` ≥ 0 voor het aantal passagiers in de `j`-de wagon van de `i`-de trein, met $0 \leq$ `i` $<$ `m` en $0 \leq$ `j` $<$ `n`. Een voorbeeld met `m = 3` en `n = 4` staat boven. Als een trein bij het eindstation stopt met de `w`-de wagon naast de stationstrap, moet een passagier uit de `j`-de wagon $10 + 20 \cdot |\texttt{w} - \texttt{j}|$ meter lopen naar deze trap.

40	90	30	30
0	50	10	40
60	40	30	70

a. (5) Schrijf een C++-functie `double loop (P,i,w)` die de gemiddelde afstand retourneert die een passagier moet lopen als de (niet-lege) `i`-de trein stopt met de `w`-de wagon naast de trap. Neem aan dat $0 \leq$ `i` $<$ `m` en $0 \leq$ `w` $<$ `n`. Het voorbeeld, met `i` $= 1$ en `w` $= 3$, levert $(50 \cdot (10 + 20 \cdot |1 - 3|) + 10 \cdot (10 + 20 \cdot |2 - 3|) + 40 \cdot (10 + 20 \cdot |3 - 3|))/100 = 32.0$ meter.

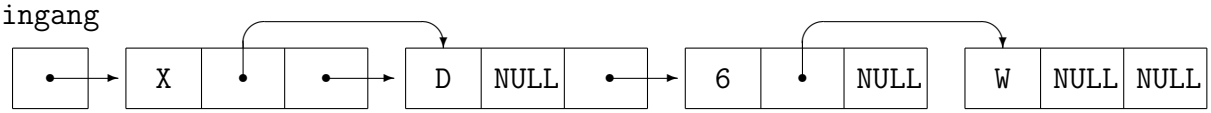
b. (10) We spreiden passagiers beter over de treinen. Er wordt een trein `i` en daarin een wagon `w` gegeven. Als hier meer dan 10 passagiers in zitten, moeten er 10 tegelijk naar een aangrenzende wagon uit dezelfde trein of naar dezelfde wagon uit vorige of volgende trein overstappen (als die er is). Zij kiezen uit deze (maximaal) vier mogelijkheden de wagon met het minste aantal passagiers; neem aan dat deze uniek is, en minder passagiers heeft dan de oorspronkelijke wagon. Schrijf hiertoe een C++-functie `verdeel (P,i,w)`. Neem aan dat $0 \leq$ `i` $<$ `m` en $0 \leq$ `w` $<$ `n`. In het voorbeeld, met `i` $= 1$ en `w` $= 1$, gaan 10 van de 50 passagiers naar de wagon die nu nog 0 passagiers heeft.

c. (10) Alle treinen moeten nu met dezelfde wagon naast de trap stoppen. Bepaal bij welke wagon dat is, als we de langste gemiddelde loop-afstand (zie **a**) zo laag mogelijk willen krijgen. Schrijf daartoe C++-functie `int beste (P,afst)` die het wagonnummer teruggeeft, en de bijbehorende langste gemiddelde loop-afstand in de `double afst` oplevert. Bij meerdere mogelijkheden: het kleinste wagonnummer.

4. (25) Gegeven is het volgende type:

```
class object { public: char info; object* volgA; object* volgB; };
```

Met behulp hiervan kan een lijst van objecten worden opgebouwd, bestaande uit vakjes met een karakter, en twee pointers. Beide pointers moeten naar het volgende object wijzen als dat er is, maar soms is één van de twee een `NULL`-pointer; de andere staat dan "goed". Een voorbeeld, met `ingang` van type `object*`:



a. (5) Schrijf een C++-functie `voegtoe (ingang,kar)` die een nieuw object netjes vooraan de lijst met ingang `ingang` toevoegt. Als `char kar` een kleine letter is, moet de bijbehorende hoofdletter hierin worden gezet, en anders `kar` zelf.

b. (5) Schrijf een C++-functie `verwijder (ingang)` die het eerste object uit de lijst met ingang `ingang` netjes verwijdert, indien dit bestaat én er een even cijfer in dit object staat.

c. (4) Schrijf een C++-functie `pasaan (ingang)` die als er ≥ 2 objecten zijn het karakter in het tweede gelijk maakt aan dat in het eerste. In het voorbeeld: `'D'` moet `'X'` worden.

d. (3) 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. (8) Stel dat sommige van de pointers terugwijzen naar een eerder object. Altijd staat minstens één van de twee pointers "goed". Schrijf een C++-functie `repareer (ingang)` die ervoor zorgt dat na afloop alle pointers naar het juiste volgende object wijzen.