

# C++ tentamen

Arnoud Visser, Tijmen Blankevoort, Auke Wiggers

February 25, 2013



UNIVERSITEIT VAN AMSTERDAM

Beantwoord alle vragen op apart tentamenpapier. Schrijf de gevraagde code zodanig dat hij compileerbaar zou zijn met elke compiler. Let op je stijl, en gebruik goede indentatie. Bij alle functies moeten de variabelen (constanten eventueel uitgezonderd) in de heading of als lokale variabele voorkomen; vul zelf headings goed in. De opgaven tellen alle vier even zwaar mee. Succes!

1. We hebben een array **A** met **n** (een *even* **const**  $> 0$ ) gehele getallen.
  - (a) Schrijf een C++-functie **bool evenveel (A,n)** die precies dan **true** oplevert als **A** evenveel positieve getallen ( $\geq 0$ ) als negatieve ( $< 0$ ) bevat.
  - (b) Schrijf een C++-functie **int epo (negpos,A,start,n)** die de array-index teruggeeft van het eerste positieve (als **negpos true** is) of negatieve (als **negpos false** is) getal in **A** vanaf (en inclusief) positie **start**. Als zo'n getal niet voorkomt, moet **n** worden geretourneerd.
  - (c) Neem aan dat **A** evenveel positieve getallen als negatieve bevat. We willen de positieve en negatieve getallen in **A** om en om krijgen, te beginnen met een positief getal. Schrijf een C++-functie **omenom (A,n)** die dit als volgt doet. Laat **i** vanaf **0** oplopen, en gebruik de functie van **b** om eventueel een posi/negatief getal te zoeken, en ruil dit met **A[i]** om.
  - (d) Stel dat de positieve getallen in **A** aan het begin staan, en de negatieve aan het eind. Hoeveel vergelijkingen tussen getallen doet het algoritme van **c** dan? (Deze worden gedaan in de aanroepen van **b**.) Tip: probeer eens **n = 10**.
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. Leg deze vier begrippen duidelijk uit.
  - (b) Gegeven een C++-programma met daarin de volgende twee functies:

```

void thierry (int a, int b, int c) {
int temp = a; a = b; b = c; c = temp; z++;
cout << "T" << a << ", " << b << ", " << c << endl; }//thierry

void henry (int a, int b, int c) {
int i, som = 0;
for ( i = 1; i <= a; i++ ) { thierry (a,b,c); som = a + b; }//for
cout << a << ", " << b << ", " << c << ", " << i << ", " << som << endl;
}//henry

```

Verder zijn de globale variabelen **u**, **x**, **y** en **z** gegeven (alle van type **int**). Voordat de functie **henry** wordt aangeroepen hebben zij de waarde 4, 1, 2 en 0, respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```

henry (u,x,y); cout << u << ", " << x << ", " << y << ", " << z << endl;

```

- (c) Als **b**, maar nu met een & ("ampersand") bij alle zes parameters van de functies.
  - (d) Wat gaat er bij **c** fout als we **som = a + b**; zouden vervangen door **a = a + b**;
  - (e) Als **c**, dus met & erbij, maar nu voor

```

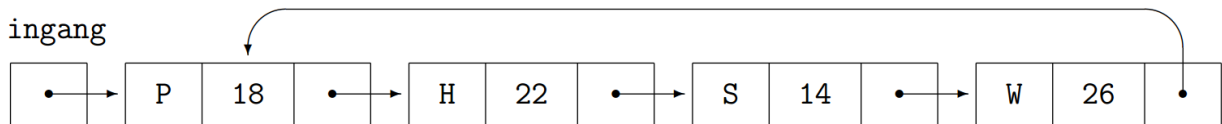
henry (z,z,z); cout << u << ", " << x << ", " << y << ", " << z << endl;

```
  - (f) Mag ergens in het C++-programma **henry (thierry (z,z,z),z,z)**; staan?
3. Gegeven is een **m** bij **n** (beide **const** > 0) array **K**; **K[i][j]** stelt het aantal op elkaar gestapelde kisten (tussen 1 en 99, alle verschillend; 0 staat voor een lege plek) op punt **(i; j)** voor. Voorbeeld van **K**:
- |    |    |    |    |   |
|----|----|----|----|---|
| 7  | 8  | 10 | 12 | 9 |
| 19 | 0  | 0  | 0  | 2 |
| 43 | 15 | 98 | 0  | 5 |
- (a) Schrijf een C++-functie **kisten (K)** die berekent hoeveel kisten er in totaal zijn in **K**.
  - (b) Een *top* is een plek in het array waarvoor geldt dat alle bestaande direct horizontale en verticale burens een aantal kisten hebben dat echt lager is. Schrijf een C++-functie **top (K,maxi,p,q)** die bepaalt hoeveel toppen er zijn in **K**, en in de variabele **maxi** de hoogte van de *hoogste top* oplevert. In **p** en **q** moeten de coördinaten van die hoogste top komen. In het voorbeeld zijn de punten met aantallen kisten **12**; **43**; **98** en **5** toppen, dus **4** stuks, en **p** wordt **2**, **q** wordt **2** en **maxi** wordt **98**. Als er helemaal geen kisten zijn moeten ze alle **0** worden.
  - (c) Een *ingang* is een plek in het array zonder kisten die aan een rand zit. In het voorbeeld is er precies één ingang: **(2; 3)**. Schrijf een C++-functie **ingang (K,i,j)** die bepaalt of **(i; j)** in **K** een ingang is. Neem aan dat  $0 < i < m$  en  $0 < j < n$ .
  - (d) Schrijf een C++-functie **tezien (K)** die bepaalt of er een ingang (zie **c**) is van waaruit de hoogste top (zie **b**) te zien is. Deze is te zien als je in dezelfde rij of kolom staat. In het voorbeeld: **true**.

4. Gegeven is het volgende type:

```
class persoon { public: char naam; int leeftijd; persoon* volg; };
```

Met behulp hiervan worden lijstjes met personen opgebouwd, met naam en leeftijd. Het veld `volg` bevat een pointer naar het volgende persoon-object; de volg-pointer van de laatste persoon wijst naar de eerste. Een voorbeeld (**ingang van type `persoon*`**):



- Schrijf een C++-functie **verwijder** (**ingang**) die het *tweede* persoon-object uit de lijst (met `ingang` van type **`persoon*`** als `ingang`) verwijdert, mits dat er is. Denk dus aan de lege lijst en de lijst met één persoon!
- Schrijf een C++-functie **voegtoe** (**ingang,name,age**) die een nieuw persoon-object (geheten **name**, leeftijd **age**) vooraan de lijst (met `ingang` van type **`persoon*`** als `ingang`) toevoegt. De pointer vanuit het laatste element hoeft niet te worden goedgezet.
- Schrijf een C++-functie **sorteer** (**ingang**) die de twee eerste *objecten* - indien aanwezig - van de lijst (met `ingang` van type **`persoon*`** als **`ingang`**) verwisselt als de namen niet op alfabetische volgorde staan. In het voorbeeld zouden de objecten met **P** en **H** erin verwisseld moeten worden. De pointer vanuit het laatste element hoeft niet te worden goedgezet. Let op het geval met twee personen.
- 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? Leg duidelijk uit.
- Schrijf een C++-functie **`persoon*` delaatste** (**ingang**) die een pointer naar de "laatste" persoon uit de lijst (met `ingang` van type **`persoon*`** als **`ingang`**) oplevert.
- Geef een of twee regels code die bij onderdeel **b** met behulp van de functie van **e** de pointer vanuit de laatste persoon weer goed zet.