Uitwerking Tentamen Programmeermethoden 1 augustus 2005

OPGAVE 1

```
a. void verwissel (int A[ ], int i, int j) {
     int temp = A[i]; A[i] = A[j]; A[j] = temp;
   }//verwissel
b. void reorganiseer (int A[ ], int n) {
     int i;
     for ( i = 0; i < n; i = i + 2 )
       if ( A[i] < A[i+1] ) verwissel (A, i, i+1);
   }//reorganiseer
c. int grootste (int A[ ], int n) {
     int i, j = 0, gr = A[0];
     for ( i = 2; i < n; i = i + 2 )
       if ( A[i] > gr ) { j = i; gr = A[i]; }//if
     cout << gr << endl; return j;
   }//grootste
d. void eenna (int A[ ], int n, int gr, int & enagr) {
     int i, enagr = gr + 1;
     for ( i = 0; i < n; i = i + 2 )
       if ( i != gr && A[i] > A[enagr] ) enagr = i;
   }//eenna
```

e. Bij b: n/2 vergelijkingen; bij c: n/2-1 en bij d: n/2-1;
samen 3 n/2 - 2.

OPGAVE 2

a. Globale variabelen gelden in het gehele programma, en worden helemaal bovenin aangemaakt. Locale variabelen gelden (tijdelijk) alleen in de functie waarin ze aangemaakt zijn.
Variabelen kunnen call by value en call by reference worden meegegeven aan een functie. Bij call by value gaat alleen de waarde van de parameter naar de functie, alwaar een locale variabele deze waarde opvangt, en er met deze locale variabele wordt verder gerekend. De oorspronkelijk variabele behoudt zijn waarde. Bij call by reference (&) gaat als het ware de variabele zelf naar de functie, en kan dan ook blijvend veranderd worden. Eigenlijk wordt het adres (de reference) doorgegeven.
Formeel: in functieheading, bijvoorbeeld x, y in int f (int x, bool y) {
Actueel: bij aanroep, bijvoorbeeld r en y in z = f (r,y);

b. romulus: 5,2,2,3 romulus: 3,1,3,2 romulus: 16,1,16,2
remus: 2,2,17,48 main: 2,6,4

c. romulus: 5,2,2,3 romulus: 1,1,1,2 romulus: 16,1,16,2
remus: 1,1,17,42 main: 1,6,42

d. remus: 1,1,1,15 main: 1,6,15
In remus wordt in q = ... 2 keer de functie romulus aangeroepen, waarbij de tweede argumenten p resp. q veranderd kunnen worden (dankzij de &); de volgorde van + ligt niet vast in C++.

e. Boven romulus moet een prototype "void remus (int p, int q);" komen. Let er op of dit ooit nog stopt.

OPGAVE 3

```
a. int aantal (int Een[ ][n], int X, int tol, int m) {
     int i, j, tel = 0;
     for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
       if ( Een[i][j] - X <= tol && X - Een[i][j] <= tol ) tel++;
     return tel;
   }//aantal
b. void bijnagelijk (int Een[ ][n], int Twee[ ][n], int tol, int m) {
     int i, j;
     for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
       if ( Een[i][j] - Twee[i][j] > tol || Twee[i][j] - Een[i][j] > tol )
         return false;
     return true;
   }//bijnagelijk
c. int herhaal (int Een[ ][n], int Twee[ ][n], int m) {
```

```
     int i = 0, j = 0, tel = 0;
     while ( i < m && j < n ) {
       if ( Een[i][j] == Twee[i][j] ) { i++; j++; }//if
       else if ( Een[i][j] < Twee[i][j] ) i++;
       else j++;
       tel++;
     }//while
     return tel;
   }//herhaal
d. int kwa (int Een[ ][n], int Twee[ ][n], int m) {
     int i, j, k, l, tel = 0;
     for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
       for ( k = 0; k < m; k++ ) for ( l = 0; l < n; l++ )
         if ( Een[i][j] == Twee[k][l] * Twee[k][l]
              || Twee[k][l] == Een[i][j] * Een[i][j] )
           tel++;
     return tel;
   }//kwa

OPGAVE 4
a. void erbij (mens* & mannen, mens* & vrouwen, int jaar, bool mv) {
     mens* nieuw = new mens; mens->leeftijd = jaar;
     if ( mv ) { nieuw->volgende = vrouwen; vrouwen = nieuw; }//if
     else { nieuw->volgende = mannen; mannen = nieuw; }//else
   }//erbij
b. void verwijder (mens* & vrouwen, mens* & mannen) {
     mens* weg;
     if ( mannen != NULL ) {
       weg = mannen; mannen = mannen->volgende; delete weg;
     }//if
     if ( vrouwen != NULL ) {
       weg = vrouwen; vrouwen = vrouwen->volgende; delete weg;
     }//if
   }//verwijder
c. void wissel (mens* mannen, mens* vrouwen) {
     int temp;
     if ( mannen != NULL && vrouwen != NULL ) {
       temp = mannen->leeftijd; mannen->leeftijd = vrouwen->leeftijd;
       vrouwen->leeftijd = temp;
     }//if
   }//wissel
d. Bij c hoeft er geen & bij (de pointers mannen en vrouwen veranderen niet),
   bij a en b moet het wel (bij b hoeven ze niet altijd te veranderen).
e. bool evenoud (mens* vrouwen, mens* mannen) {
     mens* man = mannen; mens* vrouw = vrouwen;
     while ( man != NULL && vrouw != NULL ) {
       if ( man->leeftijd != vrouw->leeftijd ) return false;
       man = man->volgende; vrouw = vrouw->volgende;
     }//while
     return ( man == NULL && vrouw == NULL );  // dan ook even lang
   }//evenoud
```