Uitwerkingen Tentamen Inleiding Programmeermethoden
maandag 2 augustus 2004

OPGAVE 1

a.
```
int grootste (double A[ ], int i, int j) {
     int gr = i;
     for ( k = i+1; k <= j; k++ )
       if ( A[k] > A[gr] ) gr = k;
     return gr;
   }//grootste
```

b.
```
double eenna (double A[ ], int n) {
     int gr = grootste (A,0,n-1), a, b;
     if ( gr == 0 )
       return A[grootste (A,1,n-1)];
     else if ( grootste == n-1 )
       return A[grootste (A,0,n-2)];
     else {
       a = grootste (A,0,gr-1);
       b = grootste (A,gr+1,n-1);
       if ( A[a] < A[b] ) return A[b]; else return A[a];
     }//else
   }//eenna
```

c.
```
void wissel (double A[ ], int i, int j) {
     double temp = A[i]; A[i] = A[j]; A[j] = temp;
   }//wissel
```

d.
```
void sorteer (double A[ ], int n) {
     int i;
     for ( i = n-1; i >= 0; i-- )
       wissel (A,grootste (A,0,i),i);
   }//sorteer
```

e. Vergelijkbaar met bubblesort (kwadratisch aantal vergelijkingen en verwisselingen).

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 int f (int x, bool y) { ...
Actueel: bij aanroep, bijvoorbeeld r en y in z = f (r,y);

b. 3 5 5 5 3 3 3 5 5 5 3 3 3 5 5 3 5 0 40 40 3 5 0

c. 3 5 5 3 5 5 3 5 5 3 5 5 3 5 5 5 3 0 40 40 5 3 5

d. 6 6 6 5 5 5 4 4 4 3 3 3 2 2 2 1 1 1 0 0 0 42 42 0

e. x * (a + b)

f. Het mag, mits eerste en tweede parameter van xkeer call by value zijn, dus zonder &. En uiteraard ergens onder de functies wisselsom en xkeer (eventueel in die laatste). Het is geen recursie (tenzij het statement in de functie xkeer zelf staat).

OPGAVE 3

a.
```
int vlakte (int Hoogte[ ][n], int m) {
     int lang = 1, i, j, lengte;
     for ( i = 0; i < m; i++ ) {
       lengte = 1;
       for ( j = 1; j < n; j++ )
         if ( A[i][j] == A[i][j-1] ) {
           lengte++; if ( lengte > lang ) lang = lengte;
         }//if
```

```
        else lengte = 1;
    }//for
    return lang;
  }//vlakte
b. int abso (int x) { if ( x > 0 ) return x; else return -x; }//abso
  int verschil (int Hoogte[ ][n], int i, int j1, int j2) {
    int totaal = 0, j;
    for ( j = j1+1; j <= j2; j++ )
      totaal += abso (Hoogte[i][j]-Hoogte[i][j-1]);
    return totaal;
  }//verschil
c. int verschil2 (int Hoogte[ ][n], int i, int j1, int j2) {
    int totaal, jup, jdown, kort = verschil (Hoogte,i,j1,j2);
    if ( i > 0 )
      for ( jup = j1; jup < j2-1 /* j2 mag ook */; jup++ )
        for ( jdown = jup+2 /* +1 mag ook */; jdown <= j2; jdown++ ) {
          totaal = verschil (Hoogte,i,j1,jup)
                   + abso (Hoogte[i][jup],Hoogte[i-1][jup])
                   + verschil (Hoogte,i-1,jup,jdown)
                   + abso (Hoogte[i][jdown],Hoogte[i-1][jdown])
                   + verschil (Hoogte,i,jdown,j2);
          if ( totaal < kort ) kort = totaal;
        }//for
    return kort;
  }//verschil2

OPGAVE 4
a. void verwissel (vak* najaar, vak* voorjaar) {
    int temp; char tempc;
    temp = najaar->code; najaar->code = voorjaar->code;
    voorjaar->code = temp;
    temp = najaar->stp; najaar->stp = voorjaar->stp;
    voorjaar->stp = temp;
    tempc = najaar->docent; najaar->docent = voorjaar->docent;
    voorjaar->docent = tempc;
  }//verwissel
b. void gooiweg (vak* & najaar) {
    vak* weg = najaar;
    najaar = najaar->volgende;  // gegeven: najaar NIET LEEG
    delete weg;
    weg = najaar;
    if ( najaar->volgende != NULL ) {
      najaar = najaar->volgende;
      delete weg;
    }//if
  }//gooiweg
c. void erbij (vak* & najaar, int c, int s, char d) {
    vak* nieuw = new vak;
    nieuw->code = c; nieuw->stp = s; nieuw->docent = d;
    nieuw->volgende = najaar; najaar = nieuw;
  }//erbij
d. void concateneer (vak* & colleges, vak* & najaar, vak* & voorjaar) {
    colleges = najaar;
    while ( najaar->volgende != NULL ) najaar = najaar->volgende;
    najaar->volgende = voorjaar;
    najaar = NULL; voorjaar = NULL;
  }//concateneer
e. Bij b, c en d moet het erbij: de lijstpointers gaan veranderen.
  Bij a maakt het niet uit: de ingangspointer verandert toch niet.
```