VANDAAG

STRUCTURES

LINKED LIST

MATRIX READING FROM FILE

STRUCTURES (1)

WAAROM

GEGEVENS VAN VERSCHILLEND TYPE ALS EENHEID BIJ ELKAAR GEVOEGD

ELK GEGEVEN APART TE BENADEREN

BELANGRIJK BIJ GEBRUIK VAN EEN COMPLEXE DATASTRUCTUUR

VOORBEELD

NAAM	STRING
STRAAT	STRING
HUISNUMMER	INTEGER
POSTCODE	STRING
WOONPLAATS	STRING
STUDENTNUMMER	INTEGER
COLL.GELD BETAALD	BOOLEAN

STRUCTURES (2)

OPBOUW VAN EEN STRUCTURE

EEN STRUCTURE BESTAAT UIT EEN AANTAL VELDEN OF "MEMBERS" DIE ELK EEN TYPE EN EEN NAAM HEBBEN

EEN STRUCTURE HEEFT ZELF OOK EEN NAAM, DIE GEBRUIKT WORDT OM DE AFZONDERLIJKE VELDEN TE BENADEREN

DEFINITIE VAN EEN STRUCTURE

DE STRUCTUUR VAN EEN STRUCTURE MOET EERST GEDEFINIEERD WORDEN

PAS DAARNA KUNNEN ER STRUCTURE VARIABELEN GEDECLAREERD WORDEN

STRUCTURES: VOORBEELD (1)

```
// definitie
// (kan zowel globaal als lokaal)
struct 3Dpoint
{ double x;
  double y;
  double z;
}


// declaratie
struct 3Dpoint hoekpuntenKubus[8];


// declaratie met initialisatie
struct 3Dpoint hoekenViervlak[4] =
        { { 0.0, 0.0, 0.0},
          { 1.0, 0.0, 0.0},
          { 0.0, 1.0, 0.0},
          { 0.0, 0.0, 1.0} }
```

STRUCTURES: VOORBEELD (2)

```
// definitie
// (kan zowel globaal als lokaal)
struct 3Dpoint
{ double x;
  double y;
  double z;
}


// declaratie
struct 3Dpoint hoekpuntenKubus[8];


// initialisatie
hoekpuntenKubus[0].x = 0.0;
hoekpuntenKubus[0].y = 0.0;
hoekpuntenKubus[0].z = 0.0;
hoekpuntenKubus[1].x = 0.0;
// maar dit kan ook in een loop ...
```

STRUCTURES: VOORBEELD (3)

```
// definities
#define MAXLEN 40
#define AANTAL 80
struct artikel
{ char   *naam;
  int    voorr;
  double prijs;
}

// declaratie
struct artikel assort[AANTAL];

// initialisatie
// eerst allocatie, dan pas vullen

for (i= ...)
{ assort[i].naam = (char *) malloc
    (MAXLEN * sizeof(char));
  assort[i].naam  = "afwasborstel";
  assort[i].voorr = 10;
  assort[i].prijs = 9.95;
}
```

LINKED LISTS

WAT IS EEN LINKED LIST

EEN AANTAL KLONTEN INFORMATIE
DIE MET ELKAAR VERBONDEN ZIJN

WAT IS ER NODIG

EEN STRUCTURE VOOR DE KLONTEN
POINTERS VOOR DE VERBINDINGEN

HOE WORDT HET GEDAAN

EEN POINTER WIJST NAAR HET
EERSTE ELEMENT

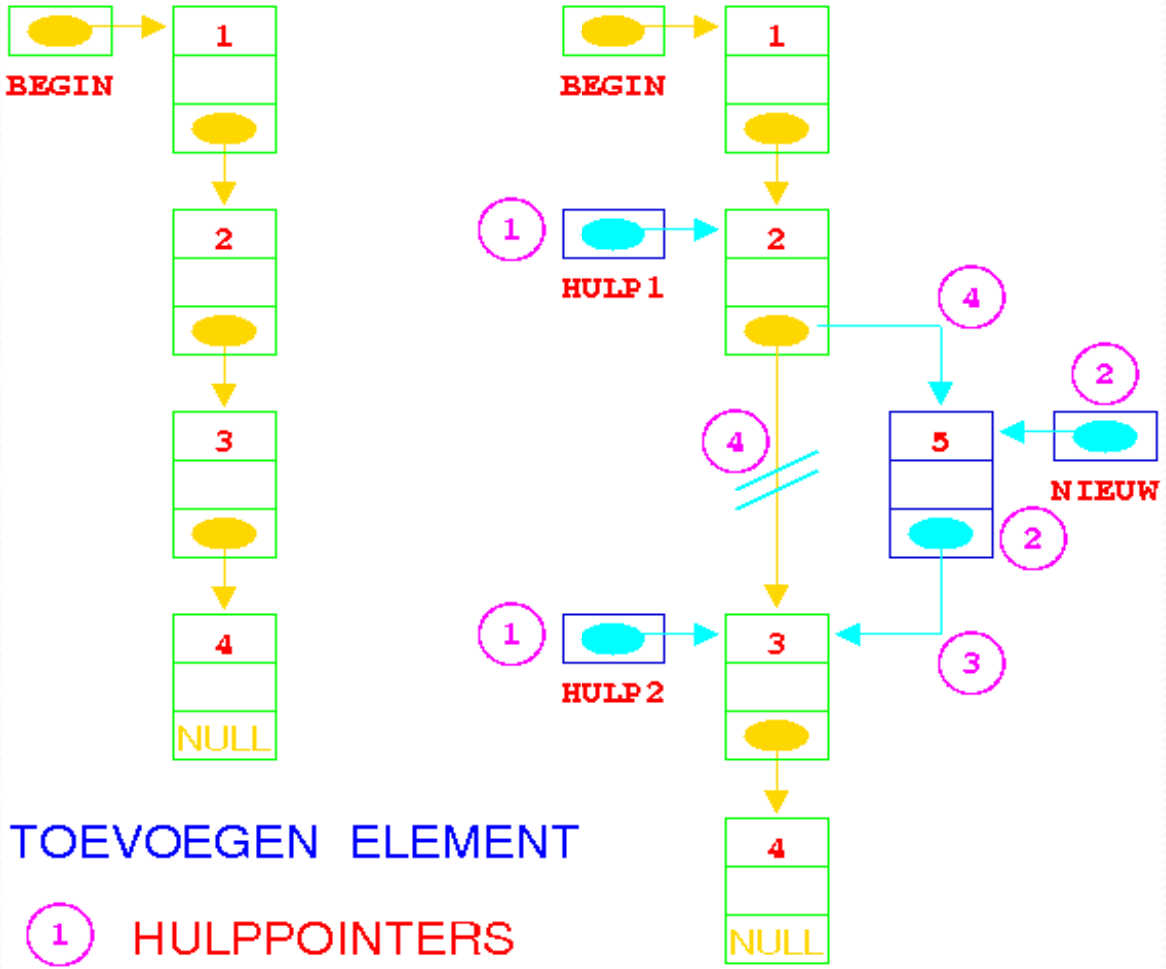
ELK ELEMENT HEEFT EEN POINTER
NAAR HET VOLGENDE ELEMENT

HET LAATSTE ELEMENT HEEFT
EEN NULL POINTER

LINKED LISTS: VOORBEELD

```
// definitie
struct point
{ double        x;
  double        y;
  struct point *volgende;
}

// declaratie
struct point *ptrPoint;
struct point *tmpPoint;

// allocatie
ptrPoint = (struct point *) malloc
              (sizeof(struct point));

// initialisatie
(*ptrPoint).x = 0.0003;
(*ptrPoint).y = 1.2345;
(*ptrPoint).volgende = NULL;

// toegang via andere pointer
tmpPoint = ptrPoint;
(*tmpPoint).x = -5.0000;
```

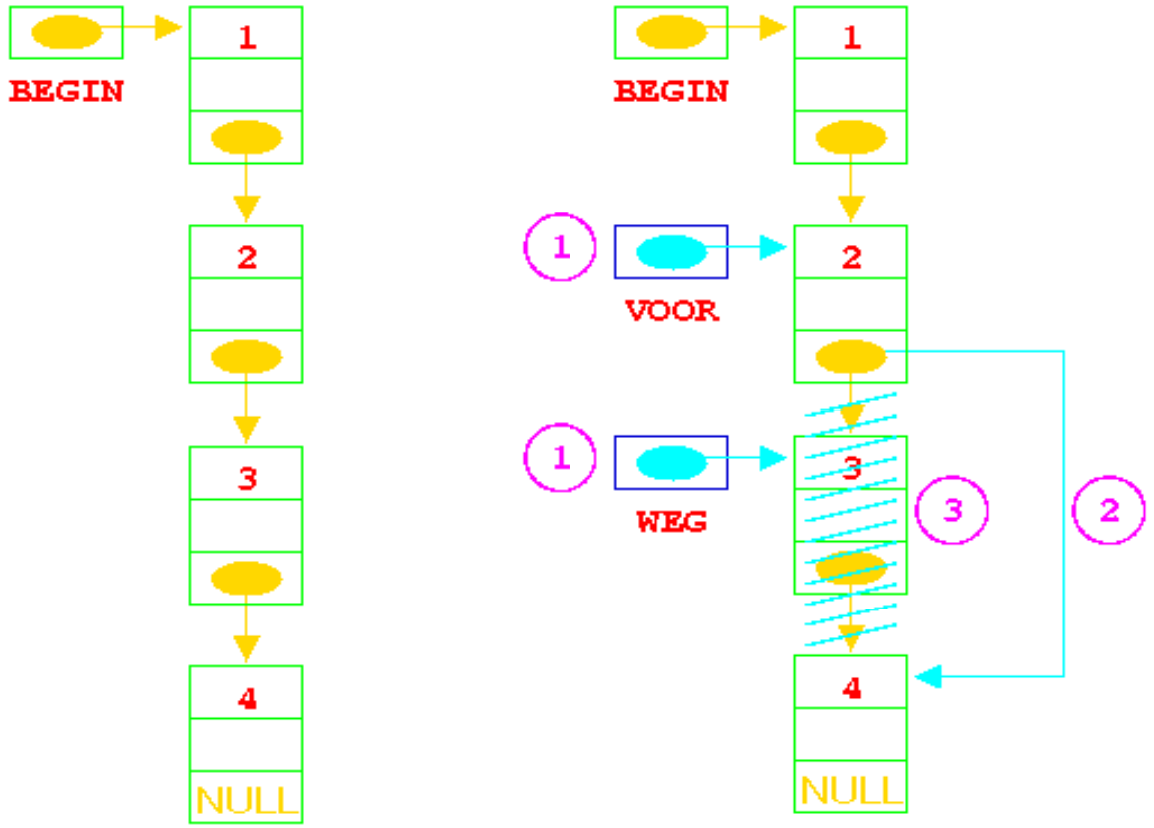
LINKED LIST: TOEVOEGEN



TOEVOEGEN ELEMENT

1. HULPPOINTERS
2. NIEUW ELEMENT MET POINTER
3. MAAK VAN NIEUW NAAR VOLGENDE
4. VERVANG VAN VORIGE NAAR NIEUW

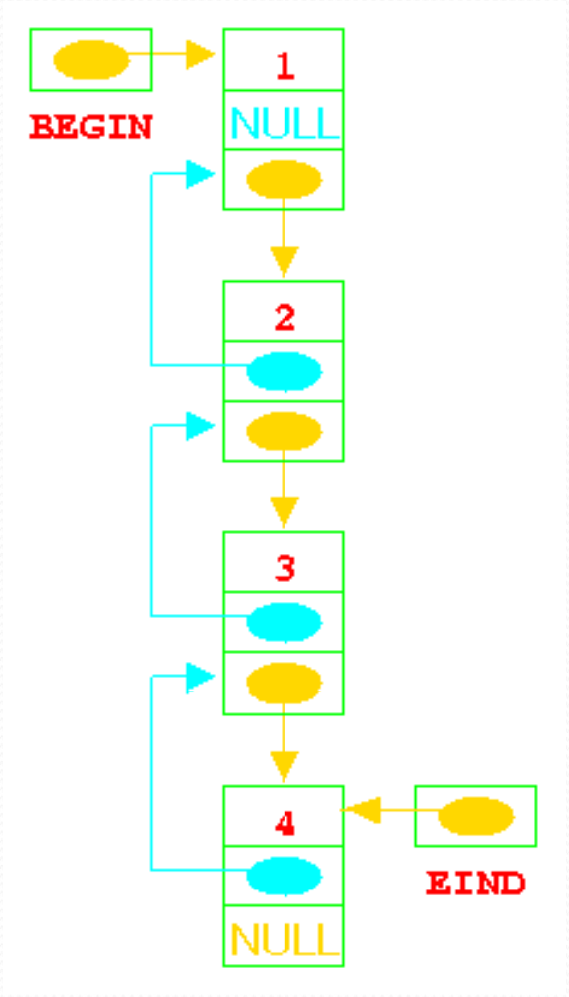
LINKED LISTS: WEGHALEN



WEGHALEN ELEMENT

1. ZET HULPPOINTERS
2. ZET POINTER VAN VORIGE OM NAAR VOLGENDE ELEMENT
3. GOOI ELEMENT WEG

DOUBLE LINKED LISTS



- Simpel voorbeeld van structures
- Structures met pointers en allocatie
- Structures en arrays
- Linked list

Algoritme voor het inlezen van de elementen een matrix uit een file

- vraag de naam van het file waaruit gelezen wordt
- open het file en test of dit succesvol was
- lees de rijen en kolommen in
- test of elke rij evenveel getallen bevat
  nee ? KLAAR met foutmelding
- KLAAR (verder met het programma)

Algoritme voor het regel voor regel inlezen van getallen uit een (geopend) file (1)

- lees regel voor regel uit het file en tel hiermee het aantal regels: Nrow
- alloceer geheugen
  - Nrow locaties voor aantal getallen per regel (int NCOL[Nrow])
  - Nrow pointers naar float pointers (float * Getal[Nrow][])
- ga naar het begin van het file

Algoritme voor het regel voor regel inlezen van getallen uit een (geopend) file (2)

- lees nu regel voor regel (i) tot einde file
  - bepaal van iedere regel het aantal getallen Ncol
    - Ncol = 0,
      - lege regel gevonden
    - Ncol > 0
      - alloceer ruimte voor Ncol getallen (float Getal[i][Ncol])
      - plaats de gevonden getallen in de array
      - bewaar voor iedere regel het aantal gevonden kolommen
- geef terug
  - Nrow
  - de gevonden getallen per regel
  - aantal gevonden getallen per regel

Algoritme voor het bepalen van aantal getallen op een regel

- verwijder eventuele lf en/of cr
- vervang eventuele TAB's door blanks
  vervang eventuele kommas door blanks
  vervang meervoudige blanks door een enkele
- verwijder begin blanks
  verwijder eind blanks
- lengte van de regel ?
  - lengte = 0
    - lege regel, KLAAR geef terug Ncol = 0
  - lengte > 0
    - tel aantal blanks: Nbl -> aantal getallen Ncol = Nbl+1
- geef de opgeschoonde regel terug en Ncol

Algoritme voor het plaatsen van de getallen op een regel in een array

- Teller = 0
- zoek tot einde van de string naar blanks
  - blank gevonden?
    - plaats nulbyte op die locatie
    - converteer string naar getal (strtod())
    - plaats het getal in de array[Teller]
    - plaats beginstring achter gevonden blank
    - Teller verhogen
  - geen blank gevonden? (laatste getal!)
    - converteer string naar getal (strtod())
    - plaats het getal in de array[Teller]
    - KLAAR

Alternatief voor algoritmes van de laatste 2 slides

Gebruik de c-functie strtok

Lees nauwkeurig de Hints en Eisen bij opdracht 12!!