F01 - Fält (*array*)

5DV149 Datastrukturer och algoritmer
Kapitel 6

Niclas Börlin
niclas.borlin@cs.umu.se

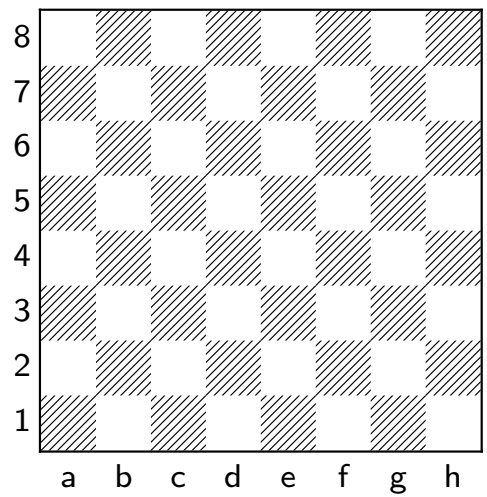
2020-01-22 Wed

Fält (1)

- En av de vanligaste fysiska datatyperna.
- Vi kommer att se på den som en abstrakt datatyp.
- Modell: Linjärt ordnade “lådor”.
  - En dimension: vektor.

0	1	2	3	4	5	6	7	8	9
49	0	58	9	0	0	0	0	18	89

  - Två dimensioner: schackbräde, matris, svartvit bild.



  - Tre dimensioner: tensor, färgbild (RGB), volym, (minecraft).

Fält (2)

- Organisation:
  - Ett n-dimensionellt fält organiserat som rätblock.
  - Tillåts innehålla odefinierade elementvärden.
- Sammansatt datatyp (innehåller andra datatyper).
  - Homogen datatyp (alla element har samma typ).
- Statisk typ:
  - Hela strukturen fixerad, inkl. storlek.
  - Elementen har bestämd plats, flyttas inte runt.
  - Kan ha dynamiska element – länkar (pekare).
- Index/koordinatvärden
  - Diskret linjärt ordnad typ (oftast heltal).
  - n-tippel, t.ex. (4) eller (e, 2) eller (x, y, z).

Gränsyta

- Ingående operationer
  - Konstruktorer (skapar ett element av typen)
    - `Create`
  - Modifikatorer (modifierar innehåll)
    - `Set-value`
  - Inspektorer (avläser innehåll eller struktur)
    - `Low`
    - `High`
    - `Inspect-value`
    - `Has-value`

Index

- Alla operationer på index flyttas till indextypen (next, previous, right-of, etc.).
- Indexgränserna
  - Egenskaper specifika för varje objekt.
  - Sätts när objektet skapas.
  - Kan inte ändras, bara avläsas.

Gränsyta till Fält

abstract datatype	**Array** (**val**, **index**)
Create (*lo, hi*: **index**)	→ **Array** (**val**, **index**)
Set-Value (*i*: **index**, *v*: **val**, *a*: **Array** (**val**, **index**))	→ **Array** (**val**, **index**)
Low (*a*: **Array** (**val**, **index**))	→ **index**
High (*a*: **Array** (**val**, **index**))	→ **index**
Has-value (*i*: index, *a*: **Array** (**val**, **index**))	→ **Bool**
Inspect-value (*i*: index, *a*: **Array** (**val**, **index**))	→ **val**

Typparametrar

- Ett Fält har två typparametrar:
  - `val` — elementvärdetypen, kan vara vilken typ som helst.
  - `index` — måste vara en n-tippel, som måste ha elementvärden som är diskret linjärt ordnade.
- Vanligast är att indextypen är tippel av heltal.
- Värdetypen varierar med tillämpningen.

Exempel: Schackbräde

- värdetyp: tecken

Qq	Queen.
Kk	King.
Rr	Rook.
Bb	Bishop.
Nn	Knight.
Pp	Pawn.

- indextyp: 2-tippel av (tecken, heltal).

lo	(a, 1)
hi	(h, 8)

Exempel: färgbild

- värdetyp: 8-bitars heltal (0-255)
- indextyp: 3-tippel av (heltal, heltal, färg), där färg är en uppräknad datatyp

  r röd
  g grön
  b blå

Konstruktion av Fält

- Fysisk datatyp i många språk.
- Inbyggda fält i C lagras som vektor.
  - N-dim Fält som 1-dim Fält.
    - "Vecklar" ut fältet.
    - Matriser lagras kolumnvis (Matlab) eller radvis.
  - N-dim Fält som Fält av Fält av `val`.
  - Fält kan konstrueras som Lista.
    - Inte så effektivt.

Gles matris

- Gles matris — ett stort antal element är odefinierade eller har värdet noll.
- Implementeras som Fält av tabell, ex. (rad, kolumn, värde).
  - Sparar utrymme.
  - Sparar tid.
- Kommer ofta från stora tekniska problem.

Tillämpningar Fält (1)

- Tekniska beräkningar
  - Geometriska transformationer (grafik, datorspel, VR, datorseende).
    - Ex: translation T, rotation R, skalning S:

$$T = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}, \; R = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; S = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

  - Linjära ekvationssystem (simulering, flödesberäkningar, autopiloter, GPS).
  - Bilder (kantdetektering, brusreducering, OCR).

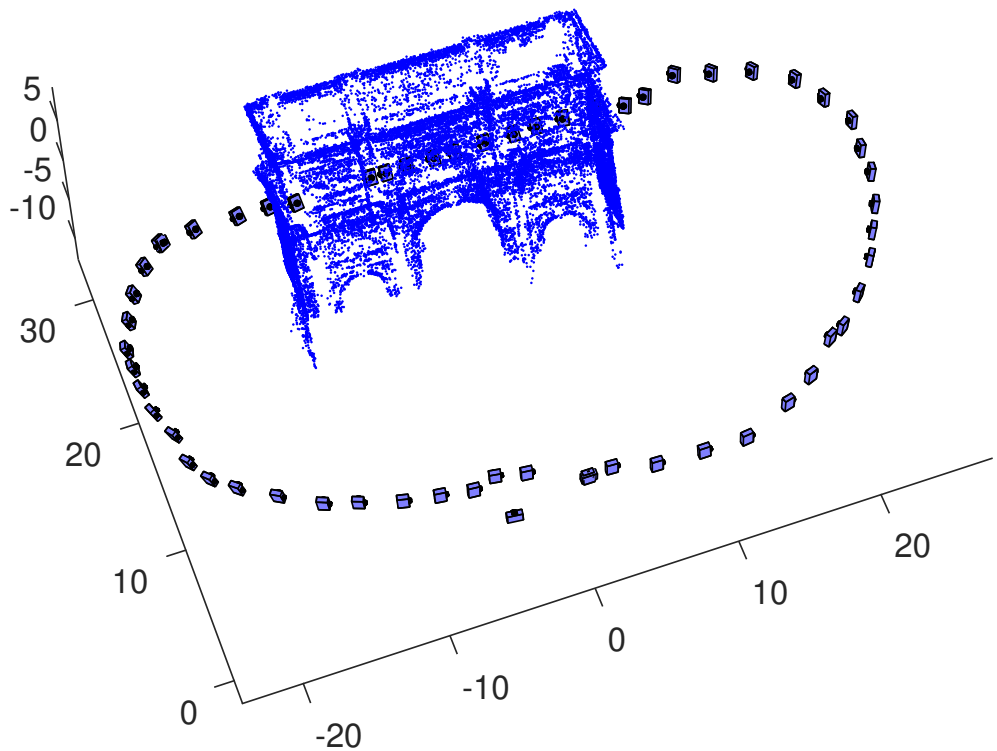
By JonMcLoone at English Wikipedia, CC BY-SA 3.0,

https://commons.wikimedia.org/w/index.php?curid=44894482

  - Spelmatriser (schack, luffarschack, minecraft).

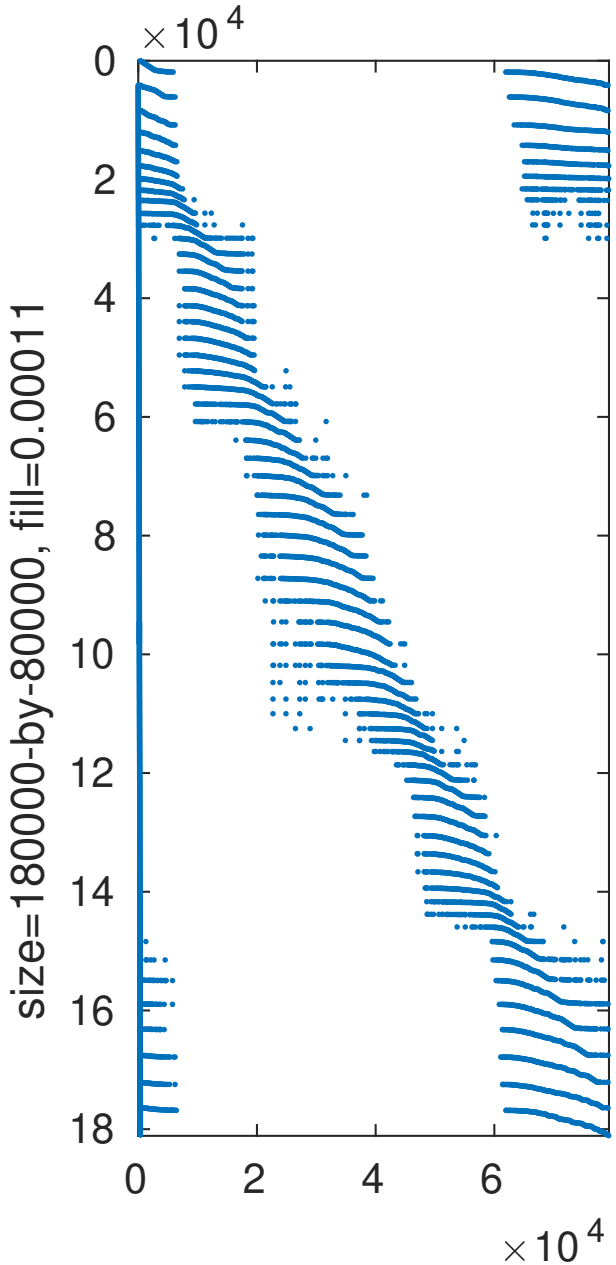
Tillämpningar Fält (2) — Fotogrammetri

- 60 bilder av ett objekt.
- 26000 3D-punkter.
- 90000 2D-mätningar.



Tillämpningar Fält (3) — Fotogrammetri

- Resulterar i en gles matrix.
- Varje rad svarar mot en 2D-mätning.
- Varje kolumn svarar mot en 3D-punkt.
- Storlek 180000-x-80000.
- 0.1 promille noll-skilda element.



Blank

Blank

Blank

Blank

Exempel 1

dcount_indexed.c

```c
#include <stdio.h>

void zero_histogram(int *hist)
{
        for (int i = 0; i < 10; i++)
                hist[i] = 0;
}

void make_histogram(const char *msg, int *hist)
{
        for (int i = 0; msg[i] != '\0'; i++)
                if (msg[i] >= '0' && msg[i] <= '9')
                        hist[msg[i] - '0']++;
}

void print_histogram(const int *hist)
{
        for (int i = 0; i < 10; i++)
                if (hist[i] > 0)
                        printf("%d: %d\n", i, hist[i]);
}
```

Exempel 1

dcount_indexed.c

```
23  int main(int argc, const char *argv[])
24  {
25          const char *msg = "090-786 68 32";
26          int n = 10;
27          int hist[n]; /* Här reserveras minnet. */
28
29          zero_histogram(hist);
30          make_histogram(msg, hist);
31          print_histogram(hist);
32
33          return 0;    /* Här återlämnas minnet. */
34  }
```

Exempel 2

dcount_dynamic.c

```c
#include <stdio.h>
#include <stdlib.h>

void zero_histogram(int *hist)
{
        for (int i = 0; i < 10; i++)
                hist[i] = 0;
}

void make_histogram(const char *msg, int *hist)
{
        for (int i = 0; msg[i] != '\0'; i++)
                if (msg[i] >= '0' && msg[i] <= '9')
                        hist[msg[i] - '0']++;
}

void print_histogram(const int *hist)
{
        for (int i = 0; i < 10; i++)
                if (hist[i] > 0)
                        printf("%d: %d\n", i, hist[i]);
}
```

Exempel 2

dcount_dynamic.c

```
24  int main(int argc, const char *argv[])
25  {
26          const char *msg = "090-786 68 32";
27          int n = 10;
28          int *hist = malloc(n * sizeof(int)); /* reserveration */
29
30          zero_histogram(hist);
31          make_histogram(msg, hist);
32          print_histogram(hist);
33
34          free(hist);                          /* återlämning */
35          return 0;
36  }
```

Exempel 3

dcount_abstract.c

```c
#include <stdio.h>
#include "array_1d_int.h"

void zero_histogram(array * a)
{
        for (int i = array_low(a); i <= array_high(a); i++)
                array_set_value(a, i, 0);
}

void make_histogram(const char *msg, array * a)
{
        for (int i = 0; msg[i] != '\0'; i++)
                if (msg[i] >= array_low(a) && msg[i] <= array_high(a))
                        array_set_value(a, msg[i],
                                        array_inspect_value(a,
                                                            msg[i]) + 1);
}

void print_histogram(const array * a)
{
        for (int i = array_low(a); i <= array_high(a); i++)
                if (array_inspect_value(a, i) > 0)
                        printf("%c: %d\n", i, array_inspect_value(a, i));
}
```

Exempel 3

dcount_abstract.c

```
26  int main(int argc, const char *argv[])
27  {
28          const char *msg = "090-786 68 32";
29          array *a = array_create('0', '9');
30
31          zero_histogram(a);
32          make_histogram(msg, a);
33          print_histogram(a);
34
35          array_kill(a);
36          return 0;
37  }
```

Övning

- Diskret/kontinuerlig
- Linjärt ordnad, m.m.
- Hur är en 2d-fält ordnat? radvis, kolumnvis?