

# EDA016 Programmeringsteknik för D

## Läsvecka 6: Vektorer

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2015

## 6 Vektorer

- Att göra denna vecka
- Vektorer
  - Exempel: Fibonacci
  - Initialisering av vektorer
  - Leta efter minimum och maximum i Array
  - Linjärsökning i Array
  - Vektorer med objektreferenser
  - Registrering

## Att göra i Vecka 6: Förstå vektorer (eng. Arrays)

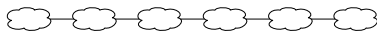
- 1 Läs följande kapitel i kursboken: 8  
Begrepp: element, vektor, Array, allokering, indexering
- 2 Gör övning 6: vektor, registrering
- 3 Träffas i samlingsgrupper och hjälp varandra
- 4 Gör Lab 5: Gissa Tal  
Om den är för lätt: bygg ut den, gärna med grafik i SimpleWindow

# Datastrukturer

En **datastruktur**:

- kan innehålla många element,
- har *ett* namn,
- och man kan komma åt de enskilda elementen.

Man kan organisera elementen på olika sätt, t.ex. som listor, träd eller grafer:

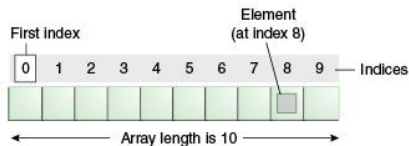
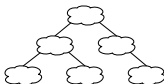
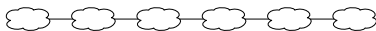


# Datastrukturer

En **datastruktur**:

- kan innehålla många element,
- har *ett* namn,
- och man kan komma åt de enskilda elementen.

Man kan organisera elementen på olika sätt, t.ex. som listor, träd eller grafer:

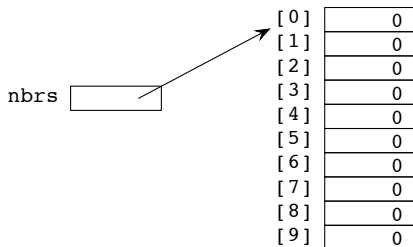


- En vektor (eng. **array**) är en vanlig datastruktur som är snabb att indexera varsomhelst i.
- Kallas även endimensionellt **fält** (eng. *one-dimensional array*).

# Vektorer (arrays) i Java

Javasyntax för att deklarerar & allokera **vektor** (eng. *array*) med 10 heltal:

```
int[] nbrs = new int[10];           //typnamn[] = new typnamn[antal];
```



- Deklarera med hak-parenteser efter typen: **int**[]
- Allokera med **new** och hakar runt antalet platser: **new int**[10]
- Index räknas från 0
- Indexera med hakparenteser: t.ex. `nbrs[3]` ger fjärde platsen
- Indexering i en tilldelningssats: `nbrs[i] = nbrs[i] + 1;`

Fyll vektorn med heltalet 42 på alla platser:

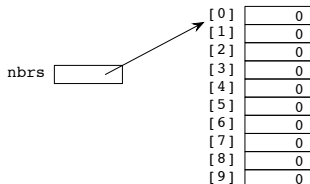
```
for (int i = 0; i < nbrs.length; i++) { // nbrs.length ger antalet platser  
    nbrs[i] = 42;  
}
```

# Vektorer: Exempel Fibonacci

Fyll vektorn med **Fibonacci**-tal:

```
1  int[] nbrs = new int[10];
2  nbrs[0] = 1;
3  nbrs[1] = 1;
4  for (int i = 2; i < nbrs.length; i++) {
5      nbrs[i] = nbrs[i - 1] + nbrs[i - 2];
6  }
```

Efter rad 1:



[0]	0
[1]	0
[2]	0
[3]	0
[4]	0
[5]	0
[6]	0
[7]	0
[8]	0
[9]	0

Övning:

- Rita innehållet i nbrs efter rad 3
- Rita innehållet i nbrs efter rad 5 för några rundor av **for**-loopen
- Vad blir innehållet i vektorn efter rad 6?

# Vektorer: Initialisering och ArrayIndexOutOfBoundsException

- Vektorer kan initialiseras med "vektor-initialiserare" (eng. *array initializers*) som skrivs med krullparenteser:

```
int[] fibFirstTen = {1, 1, 2, 3, 5, 8, 13, 21, 34, 55};
```

- Om man indexerar utanför allokerade element med negativt index eller med ett index större än length kastas ett undantag:

```
System.out.println(fibFirstTen.length); //skriver ut 10  
System.out.println(fibFirstTen[10]); //ArrayIndexOutOfBoundsException
```

- Observera att det **inte** ska vara ( ) efter length.
- Se även [Array tutorial](#).



# FibonacciArray

Se exempel `week06.FibonacciArray`:

```
1 package week06;
2
3 public class FibonacciArray {
4     private long[] fib; //array with Fibonacci numbers
5     private int computed; //number of Fibonacci numbers computed so far
6
7     public FibonacciArray(int n) {
8         fib = new long[n];
9         fib[0] = 1;
10        fib[1] = 1;
11        computed = 2;
12    }
13
14    private void compute(int n){
15        for(int i = computed; i < n; i++){
16            fib[i] = fib[i-2] + fib[i -1];
17        }
18    }
19
20    /** Return Fibonnaci-number i, counting from one */
21    public long get(int i){
22        if (i >= computed && i <= fib.length) {
23            compute(i);
24        }
25        return fib[i-1];
26    }
27 }
```

# FibonacciArrayTest

```
1 package week06;
2
3 public class FibonacciArrayTest {
4
5     public static void main(String[] args) {
6         FibonacciArray fibs = new FibonacciArray(100);
7         fibs.show();
8         long fib90 = fibs.get(90);
9         System.out.println("Fibonacci-number 90: " + fib90);
10        fibs.show();
11        long fib100 = fibs.get(100);
12        System.out.println("Fibonacci-number 100: " + fib100);
13        fibs.show();
14    }
15
16 }
```

# Räkna ut medelvärde i Array

```
public class CollaborationBonus {  
    private int[] bonus;  
    int nextFreePos = 0;  
  
    public CollaborationBonus(int numberOfGroupMembers){  
        bonus = new int[numberOfGroupMembers];  
    }  
  
    public void addIndividualBonus(int b){  
        bonus[nextFreePos] = b;  
        nextFreePos++;  
    }  
  
    public int getGroupBonus(){  
        // ÖVNING  
    }  
}
```

# Leta efter minimum i Array

```
1 package week06;
2
3 import java.util.Random;
4
5 public class MinInArray {
6     public static void main(String[] args) {
7         int n = 100;
8         Random rnd = new Random();
9         double[] rs = new double[n];
10        for (int i = 0; i < n; i++) { // populate the array
11            rs[i] = Math.round(rnd.nextDouble() * 100) / 100.0;
12            System.out.print(rs[i] + " ");
13        }
14        System.out.println();
15        double min = Integer.MAX_VALUE;
16        for (int i = 0; i < n; i++) { // Find minimum
17            if (rs[i] < min) {
18                min = rs[i];
19            }
20        }
21        System.out.println("min: " + min);
22    }
23 }
```

# Leta efter både min och max i ett svep

```
1 package week06;
2 import java.util.Random;
3
4 public class MinMaxInArrayBug {
5     public static void main(String[] args) {
6         int n = 1;
7         Random rnd = new Random();
8         double[] rs = new double[n];
9         for (int i = 0; i < n; i++){ // populate the array
10             rs[i] = Math.round(rnd.nextDouble()*100)/100.0;
11             System.out.print(rs[i] + " ");
12         }
13         System.out.println();
14         double min = Integer.MAX_VALUE;
15         double max = Integer.MIN_VALUE;
16         for (int i = 0; i < n; i++){ // Find minimum & maximum in one sweep
17             if (rs[i] < min) {
18                 min = rs[i];
19             } else if (rs[i] > max) {
20                 max = rs[i];
21             }
22         }
23         System.out.println("min: " + min + " max: " + max);
24     }
25     // CAN YOU FIND THE BUG??? Try changing line 6 to: int n = 1;
26 }
```

# Linjärsökning, pseudokod

Linjärsökning:

Börja från början och sök till vi finner; sluta direkt om vi funnit.

```
pos = "platsen för första elementet";  
while ( "fler element kvar" &&  
        "element på plats pos inte det vi söker"){  
    pos = "platsen för nästa element";  
}
```

- Funkar detta även om det inte finns några element?
- Vad blir pos om vi inte hittar något?
- Vad är **värsta fallet** för antalet rundor i while-satsen?

# Linjärsökning i heltalsvektor

```
1 package week06;
2
3 public class IntFinder {
4     private int[] xs;
5
6     public IntFinder(int n) {
7         xs = new int[n];
8     }
9
10    public void set(int index, int data) {
11        xs[index] = data;
12    }
13
14    public int get(int index) {
15        return xs[index];
16    }
17
18    /** return index of first occurrence of x, or -1 if non-existent */
19    public int linearSearch(int x) {
20        int pos = 0;
21        while (pos < xs.length && xs[pos] != x) {
22            pos++;
23        }
24        if (pos < xs.length) {
25            return pos;
26        } else {
27            return -1;
28        }
29    }
30 }
```

# Linjärsökning i heltalsvektor, testprogram

```
1 package week06;
2
3 public class DeepThought {
4     public static void main(String[] args) {
5         int n = 21;
6         IntFinder ints = new IntFinder(n);
7         for (int i = 0; i < n; i++) {
8             ints.set(i, (int) (Math.random() * 42) + 1);
9             System.out.print("ints[" + i + "] == " + ints.get(i) + " ");
10        }
11        int found = ints.linearSearch(42);
12        if (found >= 0) {
13            System.out.println("\n\nFound the Answer to " +
14                               "the Ultimate Question of Life, the Universe, and Everything!");
15        } else {
16            System.out.println("\n\nSORRY: NOT FOUND!");
17        }
18    }
19 }
20 }
```

<https://www.youtube.com/watch?v=cjEdx091RWQ>



# Vektorer med objekt (referensvariabler)

Javasyntax för att deklarera & allokera **vektor** med 10 objekt:

```
Point[] points = new Point[10];           //Typnamn[] = new Typnamn[antal];
```

Array med objektreferenser är **null** från början.

```
Point[] points = new Point[10]; // 10 referenser, alla null från början
// points[0].getX(); // ger NullPointerException
Scanner scan = new Scanner(System.in);
for (int i = 0; i < points.length; i++) {
    int x = scan.nextInt();
    int y = scan.nextInt();
    points[i] = new Point(x, y);
}
```

Prova exempel **ArrayWithObjects**

# Programexempel: Polygon

Klassen `PolygonFixedSize` beskriver en polygon med ett givet (maximalt) antal hörnpunkter. Skapa och rita en triangel:

```
PolygonFixedSize triangle = new PolygonFixedSize(3);  
triangle.addVertex(10, 10);  
triangle.addVertex(50, 10);  
triangle.addVertex(30, 40);  
triangle.draw(w);
```

# Implementering av PolygonFixedSize, del 1

```
public class PolygonFixedSize {  
    private Point[] vertices; // vektor med hörnpunkter  
    private int n;           // antalet hörnpunkter  
  
    /** Skapar en polygon som har plats för högst  
        size hörnpunkter */  
    public PolygonFixedSize(int size) {  
        vertices = new Point[size];  
        n = 0;  
    }  
  
    /** Definierar en ny punkt med koordinaterna x,y */  
    public void addVertex(int x, int y) {  
        vertices[n] = new Point(x, y);  
        n++;  
    }  
}
```

# Implementering av PolygonFixedSize, del 2

```
/** Flyttar polygonen avståndet dx i x-led, dy i y-led */  
public void move(int dx, int dy) {  
    for (int i = 0; i < n; i++) {  
        vertices[i].move(dx, dy);  
    }  
}
```

```
/** Ritar polygonen i fönstret w */  
public void draw(SimpleWindow w) {  
    if (n == 0) { return; }  
    Point start = vertices[0];  
    w.moveTo(start.getX(), start.getY());  
    for (int i = 1; i < n; i++) {  
        w.lineTo(vertices[i].getX(),  
                 vertices[i].getY());  
    }  
    w.lineTo(start.getX(), start.getY());  
}
```

# Sätt in element "mitt i" vektorn

```
/** Läger in en ny punkt med koordinaterna x,y
    Efterföljande element flyttas */
public void insertVertex(int pos, int x, int y) {
    for (int i = n; i > pos; i--) { //flytta "ner" bakifrån
        vertices[i] = vertices[i - 1];
    }
    vertices[pos] = new Point(x, y);
    n++;
}
```

Övning: Kör koden med penna och papper. Antag att det finns 2 punkter och rita vektorn vertices efter varje runda i for-loopen.

Vad händer om man försöker göra insertVertex *efter* att man har maxantalet punkter?

# Ta bort element "mitt i" vektorn

```
/** Tar bort punkten på plats pos. Efterföljande element flyttas */  
public void removeVertex(int pos) {  
    for (int i = pos; i < n - 1; i++) { //flytta "upp" efterföljande  
        vertices[i] = vertices[i + 1];  
    }  
    vertices[n - 1] = null;  
    n--;  
}
```

# "Utöka" en vektors storlek

När vektorn vertices blir full:

- 1 spara en referens till den gamla vektorn
- 2 skapa en ny, dubbelt så stor
- 3 kopiera över elementen från den gamla vektorn till den nya

```
public void addVertex(int x, int y) {  
    if (n == vertices.length) {  
        Point[] oldVertices = vertices;  
        vertices = new Point[2 * vertices.length];  
        for (int i = 0; i < oldVertices.length; i++) {  
            vertices[i] = oldVertices[i];  
        }  
    }  
    vertices[n] = new Point(x, y);  
    n++;  
}
```

Se ny version Polygon som kan utökas med fler punkter [här](#).

# Registrering

Registrering (eng. *register*) innebär att räkna antalet förekomster av ett visst värde i en sekvens av värden.

Exempel: statistik över antalet tärningskast med ett visst utfall; statistik över hur många studenter som får poäng inom olika betygsintervall på en tenta.

Steg i algoritmen:

- 1 Skapa en vektor för alla värden (eller intervall) som ska räknas.
- 2 Gå igenom alla element och öka antalet på platsen i vektorn som motsvarar värdet (eller intervallet).

Registrering ingår i övningarna 6 och 7.



# Algoritmexempel: Registrering

```
public class Test {  
    private Student[] students; // studenterna  
    private int n;              // antalet studenter  
  
    /** Skapar ett prov med plats för max studenter */  
    public Test(int max) {  
        students = new Student[max];  
        n = 0;  
    }  
  
    /** Läger till studenten s */  
    public add(Student s) {  
        students[n] = s;  
        n++;  
    }  
  
    /** Skriver ut antalet studenter som har 0,1,...,  
        50 poäng på provet */  
    public void printStatistics() { ... }  
}
```

# Olika poängintervall

0, 1, 2, ...,  
50 poäng:

```
public void printStatistics() {
    int[] count = new int[51];
    for (int i = 0; i < n; i++) {
        int index = students[i].getPoints();
        count[index]++;
    }
    // ... skriv ut antalen
}
```

0–9,  
10–19,  
20–29,  
30–39,  
40–50  
poäng:

```
public void printStatistics() {
    int[] count = new int[5];
    for (int i = 0; i < n; i++) {
        int index = students[i].getPoints() / 10;
        if (index == 5) { // om 50 poäng
            index = 4;
        }
        count[index]++;
    }
    // ... skriv ut antalen
}
```

# Oregelbundna intervall, svårändrad lösning

0–24 poäng ger betyg U, 25–34 poäng betyg 3, 35–42 poäng betyg 4, 43–50 poäng betyg 5. Antalet U-betyg registreras i `count[0]`, antalet 3-betyg i `count[1]`, osv.

```
int points = students[i].getPoints();
int index;
if (points < 25) {
    index = 0;
} else if (points < 35) {
    index = 1;
} else if (points < 43) {
    index = 2;
} else {
    index = 3;
}
count[index]++;
```

# Oregelbundna intervall, mer generell & lättändrad lösning

```
int[] limits = { 25, 35, 43, 51 };  
...  
int points = students[i].getPoints();  
int index = 0;  
while (points >= limits[index]) {  
    index++;  
}  
count[index]++;
```

Se hela exemplet här i paketet [week06.register](#)