Večkratne vrednosti

ali

tabele

VEČKRATNE VREDNOSTI

- Izpiši števila v obratnem vrstnem redu
  - zanka
    - preberi i-to število
    - shrani ga v xi
    - Povečaj i za 1
  - i = prebrano število števil
  - zanka
    - izpiši xi
    - zmanjšaj i za 1
- Kako napisati indekse?
  - `int[] x = new int[10]; // pripravimo prostor`
  - `x[0], x[1], …, x[9]`

Kdaj uporabljamo tabele

- Večje število podatkov iste vrste
- Želimo izvesti enako akcijo
  - Spreminjanje na enak način
  - Uporabljanje na enak način

Primeri deklaracij

- `int [] tabela = new int[100];`
- Prostor za 100 števil tipa int
- `String [] besedilo = new String[15];`
- Prostor za 15 nizov
- `besedilo[7]` – spremenljivka, v katero lahko shranimo niz

Primer

```
int[] x = new int [10];
i = 0;
while (i < 10) {
  pod = JOptionPane.showInputDialog("Vnesi " + (i + 1)
        + ". podatek");
  x[i] = Integer.parseInt(pod);
  i++;
}
i = 9; rezultat = "";
while (i >= 0) {
  rezultat = rezultat + "\n" + x[i];
  i--;
}
JOptionPane.ShowMessageDialog(null, rezultat,
          "Obratno", JOptionPane.PLAIN_MESSAGE);
```

TabelaObratno.java

VEČKRATNE VREDNOSTI

- Indeksi od 0 do `n - 1`
  - n je velikost (20), ki smo jo navedli v `new int[20]`
- pozor na meje
- če ne vemo, koliko elementov - zg. meja
- V Javi prostor za tabelo lahko generiramo tudi šele takrat, ko zvemo, koliko elementov ima!
- Naknadno spreminjanje velikosti tabele ni več možno

Uporaba tabel

- `double[] stevila = new double[15];`
- Prostor za 15 decimalnih števil
- `stevila[7]` – spremenljivka, v katero lahko shranimo decimalno število
- `stevila[2 * i]` – spremenljivka:
  - Katera spremenljivka - odvisno od vrednosti v `i`
  - `stevila[0]` ali `stevila[2]` ali `stevila[4]` ali `...`
  - Kaj, če je `i` denimo `10`
    - sklicujemo se na `stevila[20]`. Ta ne obstaja – napaka – program preneha delovati (se sesuje)
    - POZOR NA MEJE!!

Uporaba tabel – pogoste napake

- Imamo deklaraciji
  - `double[] stevila = new double[15];`
  - `double[] stevila1 = new double[15];`
  - `.... // stavki, s katerimi posameznim elementom obeh tabel damo vrednost`
- `stevila = 17;` // NAPAKA!!
- `System.out.println(stevila); // napaka!!`
- `If (stevila1 == stevila) ... // napaka!!`
- Uporabljamo lahko le posamezne elemente tabele in ne tabelo kot celoto!
  - Zanka za izpis, primerjanje, ...
- `stevila1 = stevila; // napaka`
- Formalno je sicer OK, a pomeni nekaj drugega
  - V tabeli `stevilo1` NI kopija elementov iz tabele `stevilo`, ampak od sedaj naprej tabeli `stevilo` in `stevilo1` označujeta ISTO tabelo!
- `stevila[2 * i]` – spremenljivka:
  - Katera spremenljivka - odvisno od vrednosti v `i`
  - Kaj, če je `i` denimo `10`
    - sklicujemo se na `stevila[20]`. Ta ne obstaja – napaka – program preneha delovati (se sesuje)
    - POZOR NA MEJE!!

Primeri deklaracij

- `int[]  tabela = new int[100];`
- Prostor za 100 števil tipa int
- `double[] stevila = new double[15];`
- Prostor za 15 decimalnih števil
- `int[] x;`
- Le napoved, da je `x` tabela. Kako velika, še nismo povedali!
- `x = new int[199];`
- Šele sedaj pripravili prostor za 199 celih števil

Primeri deklaracij

- `String[]  besede = new String[17];`
- Prostor za 17 nizov
- `double[] stevila;`
- Napoved, da ima spremenljivka `stevila` indekse in ni običajna spremenljivka
- `stevila = new int[19];`
- Šele sedaj pripravili prostor za 19 decimalnih števil!
- `stevila = new int[2 * n + 1];`
- Tudi OK, če vemo, koliko je `n` (če je v spremenljivki `n` shranjena celoštevilska vrednost).

Analiza metov kocke

- Vrzimo kocko n krat in štejmo šestice, petice, ..., enke. Ugotovimo, koliko se število razlikuje od teoretične verjetnosti 1/6.
- Podatki
  - Število metov
- Kako
  - N x izvedemo zanko
  - Vržemo kocko
  - Povečamo ustrezen števec
- Zadnjič: 6 števcev, 6 x pogojni stavek, ...

Števci

- Za števce metov bomo uporabili tabelo!
- `int[] kocka = new int[6];`
- `kocka[0], kocka[1], kocka[2], kocka[3], kocka[4], kocka[5]`
- V `kocka[0]` bomo šteli, kolikokrat smo vrgli 1, v `kocka[3]` kolikokrat smo vrgli 4, ...
- Nastavimo števce na 0
- ```
  i = 0;
  while (i < 6)
  {  kocka[i] = 0;
  }
  ```
  ```

Beleženje metov

```
met  = 1 + (int)(6 * Math.random());
if (met == 1)
{   kocka[0] = kocka[0] + 1;
}
if (met == 2)
{   kocka[1] = kocka[1] + 1;
}
...
```

- S tem nismo pridobili kaj veliko – skoraj vseeno, če bi imeli števce `st1`, `st2`, ... (brez tabele)
- A beleženje lahko napišemo tudi tako
  - `kocka[met - 1] = kocka[met - 1] + 1;`
  - Pogojni stavki niso potrebni
  - Če je bil met 4, se je povečala vrednost v `kocka[3]`, ki šteje vržene štirice, ...

Glavna zanka

```
i = 1;
while (i <= 1000000)
{
  // vržemo kocko
  met  = 1 + (int)(6 * Math.random());
  // povečamo ustrezen števec
  kocka[met - 1] = kocka[met - 1] + 1;
   i = i + 1;
}
// izpis
odg = "";
i = 1;
while (i <= 6)
{
  odg = odg + i + " smo vrgli " + kocka[i -1] + "krat.\n";
  i = i + 1;
}
```

AnalizaKocke.java

Polnjenje tabele

- Tabelo velikosti n napolnimo z naključnimi števili med a in b
- `int[] nak_tabela; // velikost bomo povedali kasneje!`
- Preberemo velikost (`showInputDialog`, `parseInt`, ...)
- Določimo tabelo
  - `nak_tabela = new int[n];`
  - Tabela, velikosti `n`
  - Če smo za `n` prebrali `10`, tabela za 10 števil, če smo prebrali `1345`, tabela za 1345 števil!

Polnjenje tabele – glavna zanka

```
i = 0;
while (i < n)
{
  // izberemo naključno št. med a in b
  naklj_st = (int)((b - a + 1) * Math.random() + a);
  // shranimo v tabelo
  nak_tabela[i] = naklj_st;
  i = i + 1;
}
```

NakTabela.java

Album sličic

- Anže se je odločil, da bo zbiral sličice. Tiste o živalih. Kupiš čokoladko, v ovitku je skrita sličica živali in potem to sličico nalepiš v album.
- Mene, kot starša, pa zanima, koliko bo zaradi tega obremenjen družinski proračun, torej, koliko čokoladic bom moral kupiti, da bo album poln!

Ideja

- Izračunajmo!
  - Pa se še spomnimo toliko matematike?
- S pomočjo računalnika simulirati nakupovanje.
- Polnjenje albuma izvesti velikokrat in določiti povprečje.
- Če bomo to izvedli dovolj-krat, se matematični in "statistični" izračun ne bosta veliko razlikovala!

Predpostavke

- Proizvajalci so pošteni
  - Vse sličice nastopajo enako pogosto
- Anže ne pozna nikogar, ki bi zbiral iste sličice in nima možnosti menjave.

Kupovanje

- Enostavno
  - Izberemo naključno število med 1 in velikostjo albuma (številom sličic)
- `(int)(Math.random() * velAlbuma) + 1`

Album

- Sličica – posamezni element tabele
- Tabela tako velika kot album (kot je število sličic)
- Zanima nas le, če sličica je, ali ni
  - `boolean[] album = new boolean[velAlbuma];`
- `album[5]` ... ali imamo sličico 5
- Ni ok ... Zadnje sličice ne moremo dobiti
  - indeksi od 0 do `velAlbuma - 1`
- Zato raje
  - `boolean[] album = new boolean[velAlbuma + 1];`
- Na `album[0]` pa kar pozabimo

Osrednji del

- Kupimo čokolado
  - `slicica = (int)(velAlbuma * Math.random()) + 1;`
- Če sličice še nimamo, jo "nalepimo"
  - `album[slicica] = true;`
- Ponavljamo, dokler album ni poln!
- Kako vedeti, da je poln
  - Vsakič pregledati, če so nalepljene že vse sličice (če ni nobena vrednost `album[i] false`)
  - predolgo
  - Pomniti, koliko sličic nam še manjka

Osrednji del

```
while (kolikoManjka > 0)
{
    slicica = (int)(velAlbuma * Math.random()) + 1;
    // kupil sem sličico
    kolikoKupil = kolikoKupil + 1;
    if (!album[slicica]) // nimam je še!
    {
        kolikoManjka = kolikoManjka - 1;
        album[slicica] = true; // nalepim;
    }
}
```


Album.java

```
import javax.swing.*;
import java.applet.*;
import java.awt.*;

public class Album extends Applet
{
   // koliko sličic je potrebno kupiti, da napolnimo
    album


   public void init()
   {
      boolean[] album; // album s sličicami - true
    imam, false niamm sličice
      int velAlbuma; // velikost albuma
      int slicica; // katero sličico sem kupil
      int kolikoManjka; // koliko sličic mi še manjka
      int kolikoKupil; // zanima me, koliko sličic
    sem kupil

      int i;

      velAlbuma =
    Integer.parseInt(JOptionPane.showInputDialog
        ("Koliko sličic je v albumu"));
      album = new boolean[velAlbuma + 1];
      // da ne bomo šteli od 0 dalje!

      // na začetku je album prazen

      kolikoManjka = velAlbuma;
      kolikoKupil = 0;
      i = 1;
      while (i <= velAlbuma)
      {
         album[i] = false;
         i = i + 1;
      }

      // dokler ne napolnim albuma
      JOptionPane.showMessageDialog(null, "Po nakupih!");
      while (kolikoManjka > 0)
      {
         slicica = (int)(velAlbuma * Math.random()) + 1;
         // kupil sem sličico
         kolikoKupil = kolikoKupil + 1;
         if (!album[slicica]) // nimam je še!
         {
            kolikoManjka = kolikoManjka - 1;
            album[slicica] = true; // nalepim;
         }
      }
      // album je poln
      JOptionPane.showMessageDialog(null, "Kupil sem " +
    kolikoKupil + " sličic");
   }
}
```

Pokemon album

- Tu ni čokoladic, ampak je v zavojčku 6 sličic
- Kupovanje malo drugače:
- Namesto

```
slicica = (int)(velAlbuma * Math.random()) + 1;
```

- damo
-

```
// kupili smo nov zavojček. V njem je 6 sličic!
i = 1;
while (i <= 6)
{
   slicica = (int)(velAlbuma * Math.random()) + 1;
   // kupil sem sličico
   if (!album[slicica]) // nimam je še!
   {
      kolikoManjka = kolikoManjka - 1;
      album[slicica] = true; // nalepim;
   }
   i = i + 1;
}
```

AlbumPokemon.java

Izboljšava

- V zavojčku so različne sličice
- Kako generirati 6 različnih števil!
- Ideja:
  - Izberemo indeks sličice
  - Na začetku je številka sličice enaka indeksu
  - Prvič so kandidati mesta(indeksi) od 1 do velAlbuma
  - Drugič so kandidati mesta od 1 do velAlbuma - 1
  - tretjič ...

Generiranje različnih

- na mesto izbire damo "zadnjo" sličico, da bo v naslednjem koraku OK saj zadnje mesto ne bomo upoštevali!
- Sličica na mestu izbire pa tako ne bo prišla več v poštev
- 1 2 3 4 5 ..... Izbiramo med 1 – 5. Denimo da izberemo 2
- 1 5 3 4 2 .... Izbiramo med 1 – 4. Denimo, da izberemo 4
- 1 5 3 4 2 .... Izbiramo med 1 – 3. Denimo, da izberemo 2 (torej dejansko sličico 5!)
- 1 3 5 4 2 ... Konec. Izbrali smo torej 2, 4 in 5!

Generiranje različnih

```
i = 1;
int[] slike = new int[velAlbuma+1];
while (i <= velAlbuma) {
   slike[i] = i; // ta tabela označuje številko sličice
   i = i + 1;
}
i = 1;
while (i <= 6)  {
   // Izberemo iz tabele slik sličico.
   // Prvič so kandidati mesta od 1 do velAlbuma
   // drugič so kandidati mesta od 1 do velAlbuma - 1, tretjič ...
   izbira = (int)((velAlbuma - (i - 1))* Math.random()) + 1;
   slicica = slike[izbira];
   slike[izbira] = slike[velAlbuma - (i - 1)];
        // na mesto izbire damo "zadnjo" sličico, da bo v naslednjem koraku OK
        // saj zadnje mesto ne bomo upoštevali! Sličica na mestu izbire pa
        // tako ne bo prišla več v poštev
   if (!album[slicica]) // nimam je še!     {
      kolikoManjka = kolikoManjka - 1;
      album[slicica] = true; // nalepim;
   }
  i = i + 1;
}
```

AlbumPokemonRazlicno.java

Več nakupov

- Doslej izvedli le en nakup
- Da bo rezultat smiselen – vso polnitev ponoviti velikokrat
- Celoten program damo v zanko
- Ta se izvede denimo 1000x
- Računamo povprečje
  - Vsakič k vsoti prištejemo število potrebnih nakupov za tekočo polnitev albuma
  - Na koncu vsoto delimo s 1000

Več programov

- AlbumPokemonRazlicnoVeckrat.java
  - album s Pokemoni (zavojček s 6 različnimi sličicami), kjer povemo, koliko simulacij bomo izvedli
- AlbumGrafika1.java
  - "klasični" album z grafičnim prikazom polnitve
- AlbumGrafika2.java
  - "klasični" album z grafičnim prikazom polnitve in označevanjem podvojenih (oz. pomnoženih)
- AlbumGrafika3.java
  - "klasični" album z grafičnim prikazom polnitve in barvnim prikazom kolikokrat dobimo posamezno sličico
  - Uporabljen switch stavek kot nadomestilo za več if stavkov!

Zanimiva napaka

- Zakaj se tale program sesuje?

Tabela.java

```
int i = 0;

   while (i < n)
   { // izris naključnih stolpcev
     if (tabela[i] < 0.90 * povprecje) // izrise stevila, ki so za 10 % manjsa od povprecja
     {  g.setColor(Color.green);
        g.fillRect(x + i * (sirina + razmik), y, sirina, tabela[i]);
        i = i + 1;
     }
     if (tabela[i] > 1.10 * povprecje) // izrise stevila, ki so za 10 % vecja od povprecja
     {
         g.setColor(Color.red);
         g.fillRect(x + i * (sirina + razmik), y, sirina, tabela[i]);
         i = i + 1;
     }
     if ((tabela[i] <= 1.10 * povprecje)&&(tabela[i] >= 0.90 * povprecje))
     {
         g.setColor(Color.yellow); // izrise stevila, so +/- 10 % od povprecja
         g.fillRect(x + i * (sirina + razmik), y, sirina, tabela[i]);
         i = i + 1;
     }
   }
```