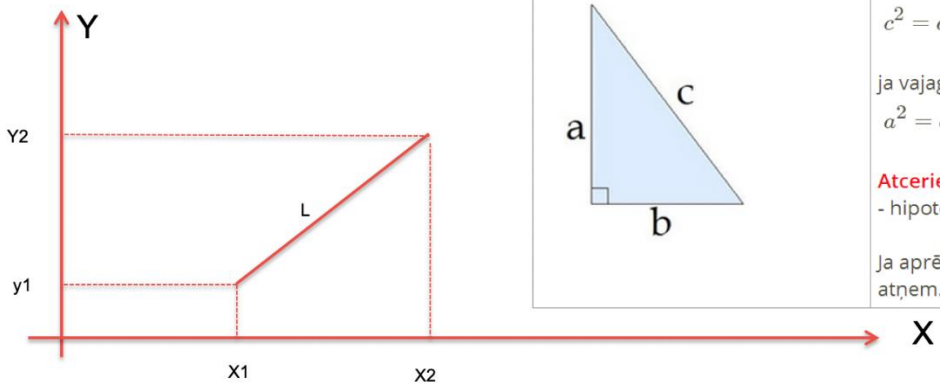
TUVINĀTIE APRĒĶINI
39.nodarbība

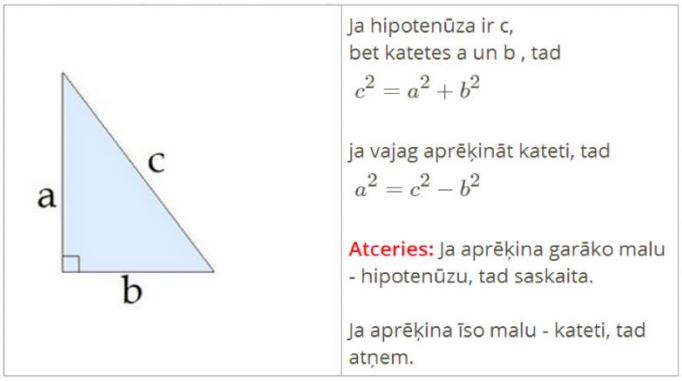
TEORIJA

- Šodien aplūkosim dažus ģeometrisko aprēķinu algoritmus, kur mēs izmantojam formulas. Aprēķināsim laukumu trijstūrim un izliektam daudzstūrim.
- Pēc tam pievērsīsimies neregulāru laukumu aprēķinu metodēm.

NOGRIEŽŅA GARUMS

Cik garš ir nogrieznis ar galapunktu koordinātām (x1,y1) un (x2,y2)?





Izmantosim Pitagora teorēmu: L = kvadrātsakne no ((x2-x1)² + (y2-y1)²)

TRIJSTŪRA LAUKUMS AR HĒRONA FORMULU

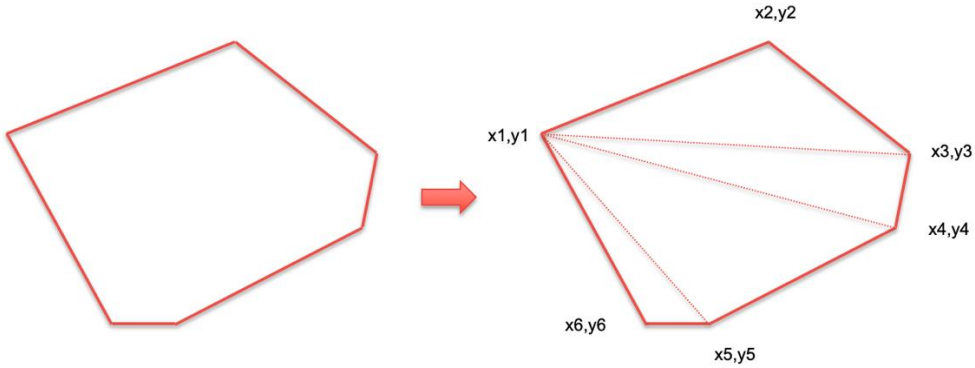
Ja a, b un c ir trijstūra malas, tad laukums

$$S_\Delta = \sqrt{p\,(p-a)\,(p-b)\,(p-c)}$$

$$p - pusperimetrs$$
$$p = \frac{a+b+c}{2}$$

IZLIEKTA DAUDZSTŪRA LAUKUMS

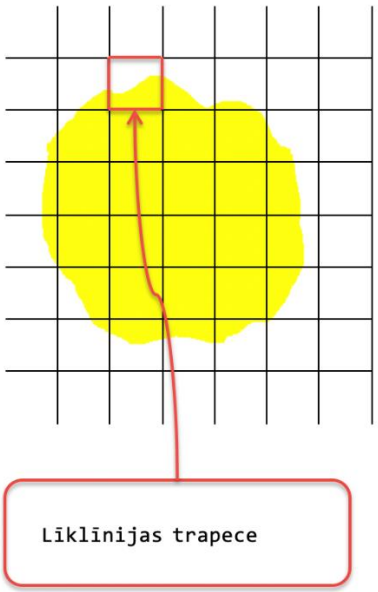
- Izliekta daudzstūra gadījumā no vienas virsotnes uz citām novelkam nogriežņus un tā sadalām daudzstūri trijstūros.
- Aprēķinām katra trijstūra laukumu. Summējam trijstūru laukumus.



NEREGULĀRAS FORMAS FIGŪRU LAUKUMI

- Taisnstūru, trijstūru, elipšu un citu svarīgu figūru laukuma aprēķināšanai izmantojam gatavas formulas.
- Bet pankūku laukuma formulu nav, un arī pašas pankūkas katrreiz iznāk atšķirīgas.
- Tomēr dzīvē vajag aprēķināt gan dažādas formas zemes un ūdens platību laukumus, gan ādas, lapu un citu materiālu laukumus.
- Šādām vajadzībām izmanto laukumu tuvinātā aprēķina metodes.

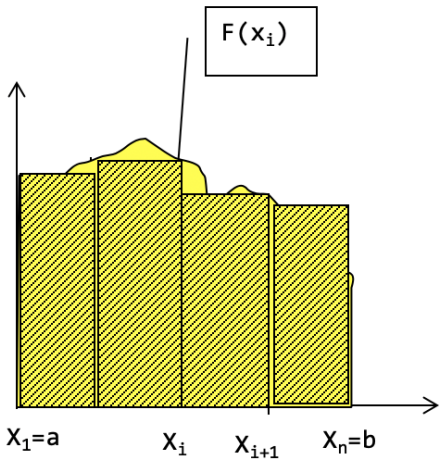
LĪKLĪNIJAS TRAPECE



Daudzas no tuvinātajām metodēm vispirms paredz figūras laukuma sarūtošanu, t.i., figūru sadala taisnstūros, kam viegli aprēķināt laukumu. Attēlā redzamo figūru noklāj daži pilni dzelteni kvadrātiņi, bet dažiem 3 malas ir taisnas, bet viena līkumota. Šādu figūru ar 3 taisnām un vienu liektu malu sauc par līklīnijas trapeci.

Parasti līklīnijas formula nav zināma. To var mēģināt tuvināti noteikt, bet mēs apskatīsim tādus gadījumus, kur funkcija jau ir zināma.

TAISNSTŪRU METODE

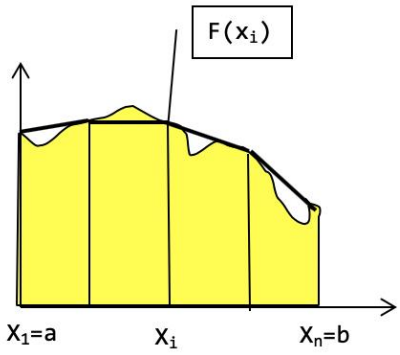


Algoritms

```
S←0
i←1
pamats←(b-a)/n
kamēr i<n
      augstums←f(pamats*i)
      S←S+pamats*augstums
```

Metode paredz, ka līklīnijas trapeci noklāj ar n taisnstūriem un aprēķina to laukumu summu. Jo n lielāks, jo taisnstūru laukumi mazāki un to summa precīzāk atbilst līklīnijas trapeces laukumam. Katra taisnstūra laukumu nosaka divas tā malas. Horizontālā mala visiem taisnstūriem ir vienāda, tās garums ir atkarīgs no n jeb tā, cik taisnstūros dala figūru. Vertikālā mala ir vienāda ar ierobežojošās funkcijas vērtību punktā x_i.

TRAPEČU METODE



Algoritms

```
S←0
i←1
pamats←(b-a)/n
kamēr i<n
      viduslīnija←(f(pamats*i)+f(pamats*(i+1))/2
```

S←S+pamats*viduslīnija

i←i+1

Līdzīga taisnstūru metodei, tikai taisnstūru vietā figūru noklāj ar trapecēm.

ŠPIKERIS PALĪGRĪKIEM

Java kods skaitļa x kāpināšanai kvadrātā:

```
double rezultats = Math.pow(x,2);
```

Rezultāts šim darbībām ir ar tipu double

Java kods kvadrātsaknes vilkšanai no x:

```
double rezultats = Math.sqrt(x);
```

Rezultāts šim darbībām ir ar tipu double

PĀRBAUDĪSIM PALĪGRĪKUS

Ievietojam main metodē kodu:

```
public static void main(String[] args) {
   double x = 9;
   double rezultats = Math.pow(x,2);
   System.out.println(x + " kvadrata r " + rezultats);
   double rezultats2 = Math.sqrt(x);
   System.out.println("kvadratsakne no " + x + " ir " + rezultats2);
}
```

- Lai programmu palaistu, ar labo peles pogu uzklikšķinām uz klases «Aprekini», Run as -> Java application
- Uz konsoles pārbaudām, vai rezultāts ir tas, ko sagaidām.

NOGR.GARUMS – JAVA KODS

Izveidojam metodi, kas aprēķinās nogriežņa garumu:

```
public static double nogrGarums (double x1, double y1, double x2, double y2) {
      double garums = Math.sqrt
      (Math.pow((x2-x1),2) + Math.pow((y2-y1),2));
      return garums;
}
```

Vai atpazīstat java kodā Pitagora formulu?

l = kvadrātsakne no $((x2-x1)^2 + (y2-y1)^2)$

Mēs šo metodi izveidojām, lai izmantotu tālākajos aprēķinos.

NOGR.GARUMS – TESTĒJAM I

Izveidojam metodi, kas testēs metodi nogrGarums:

```
public static void testNogrGarums () {
   double rezultats = nogrGarums (0,0,0,10);
   if (rezultats == 10) {
      System.out.println("Pareizi! Garums ir " + rezultats);
   } else {
      System.out.println("Nepareizi! Vajadzeja but 10!");
   }
   // pārbaudam, kā izvadās daļskaitlis
   rezultats = nogrGarums (0,0,10,10);
   System.out.println("otrs rezultats ir " + rezultats);
}
```

NOGR.GARUMS – TESTĒJAM II

No main metodes izsaucam testNogrGarums() :

```
public static void main(String[] args) {
   testNogrGarums();
}
```

Palaižam programmu: ar labo peles pogu uzklikšķinām uz klases «Aprekini», Run as -> Java application

- Uz konsoles pārbaudām, vai rezultāts ir tas, ko sagaidām. Jāsanāk apmēram šādam izvadam:

  Pareizi! Garums ir 10.0

  otrs rezultats ir 14.142135623730951

METODI TESTĒJAM AR CITU METODI

Vienas programmas testēšana ar otru programmu ir industrijā izplatīts paņēmiens. Tam ir daudzi plusi:

- Kad tests vienreiz uzrakstīts, to var pēc vajadzības atkārtot, lai pārliecinātos, vai kaut kas nav «salūzis».
- Var automātiski pārbaudīt, vai rezultāts ir tāds, kādu sagaidām.

Alternatīva ir testēt «ar roku», piemēram, ievadot datus no konsoles. Tomēr, ja ievaddati ir sarežģīti, ar roku tos ievadot, var kļūdīties. Tad darbs jāsāk no gala. Savukārt, ja tests ir ieprogrammēts, kļūdu var izlabot un testu atkārtot.

TRIJSTŪRA LAUKUMS – JAVA KODS

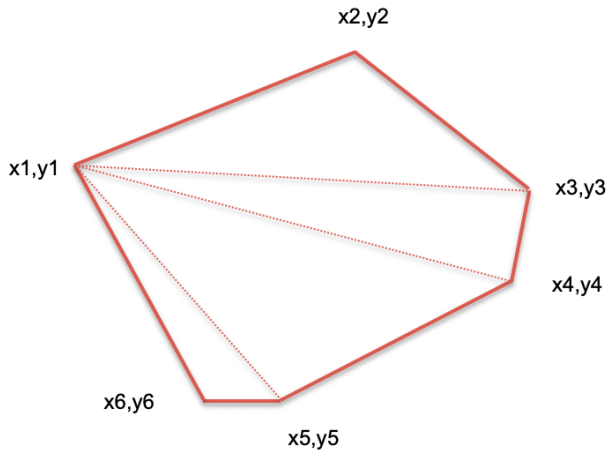
Izveidojam metodi, kas rēķinās trijstūra laukumu:

```
public static double trijstLaukums(double x1, double y1, double x2, double y2, double x3, double y3) {
   double mala1 = nogrGarums(x1, y1, x2, y2);
   double mala2 = nogrGarums(x1, y1, x3, y3);
   double mala3 = nogrGarums(x2, y2, x3, y3);
   double pusperimetrs = (mala1 + mala2 + mala3) / 2;
   double laukums = Math.sqrt(pusperimetrs *
     (pusperimetrs - mala1) *
     (pusperimetrs - mala2) *
     (pusperimetrs - mala3));
   return laukums;
}
```

Vai atpazīstat java kodā Hērona formulu?

$$S_\Delta = \sqrt{p\,(p-a)\,(p-b)\,(p-c)}$$

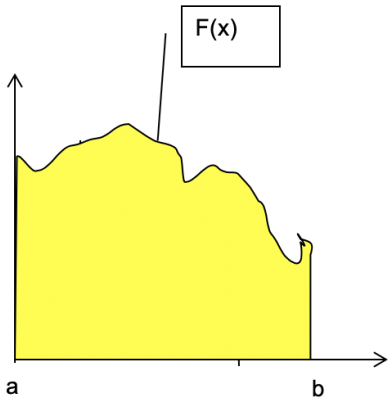
IZLIEKTA DAUDZSTŪRA LAUKUMS



Kods zīmējumā redzamā daudzstūra laukuma aprēķināšanai būs šāds:

```
laukums = trijstLaukums(x1,y1,x2,y2,x3,y3);
laukums = laukums + trijstLaukums(x1,y1,x3,y3,x4,y4);
laukums = laukums + trijstLaukums(x1,y1,x4,y4,x5,y5);
laukums = laukums + trijstLaukums(x1,y1,x5,y5,x6,y6);
```

TUVINĀTIE APRĒĶINI

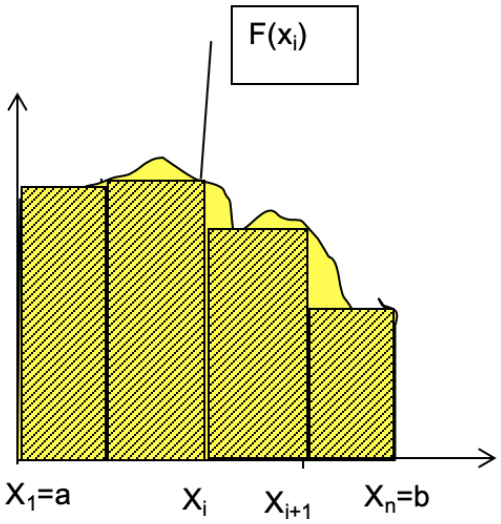


Turpmākos uzdevumos mērīsim laukumu līklīnijas trapecei. Trapeci definē šādi lielumi:

- a, b – tie mums būs parametri ar tipu double
- F(x) – kāda funkcija, ko var aprēķināt ar formulu. Mūsu vingrinājumam pieņemsim ka F(x) = sin(x) + 2. Javas sintaksē tas ir:

```
double vertiba = Math.sin(x) + 2;
```

TUVINĀTIE APRĒĶINI: TAISNSTŪRU METODE I



Algoritms

```
s←0
i←1
pamats←(b-a)/n
kamēr i<=n
      augstums←f(pamats*i)
      s←s+pamats*augstums
```

Metodes parametri: double a, double b, int n

Atgriežamā vērtība (laukums s): double

```
public static double taisnstMetode(double a, double b, int n) {
   double s = 0;
   double pamats = (b-a)/n;
   for (int i=1; i <= n;i++) {
      double augstums = Math.sin(pamats*i) + 2;
      s = s + pamats*augstums;
   }
   return s;
}
```

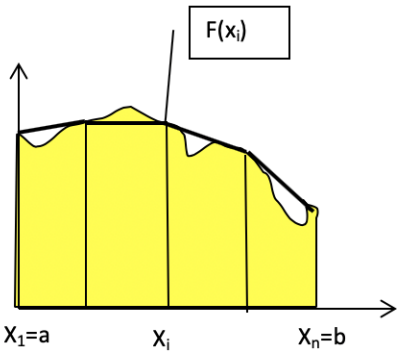
TUVINĀTIE APRĒĶINI: TAISNSTŪRU METODE II

Izveidojam metodi, kas testēs: taisnstMetode()

```
public static void testTaisnstMetode() {
   System.out.println("Taisnsturu metodes tests");
   double laukums = taisnstMetode(1,10,4);
   System.out.println("n=4, laukums=" + laukums);
   laukums = taisnstMetode(1,10,10);
   System.out.println("n=10, laukums=" + laukums);
   laukums = taisnstMetode(1,10,100);
   System.out.println("n=100, laukums=" + laukums);
}
```

- Iekopējiet kodu un notestējiet aprēķinu. Neaizmirstiet izsaukt **testTaisnstMetode()** no main() metodes.
- Palieliniet precizitāti (palielinot n) un pavērojiet, kā mainās rezultāts.

TUVINĀTIE APRĒĶINI: TRAPEČU METODE



Algoritms

```
S←0
i←1
pamats←(b-a)/n
kamēr i<n
      viduslīnija←(f(pamats*i)+f(pamats*(i+1))/2
      S←S+pamats*viduslīnija
      i←i+1
```

- Uzprogrammējiet un notestējiet trapeču metodi paši, pēc iepriekšējā parauga.
- Salīdziniet rezultātu ar taisnstūru metodi.

TUVINĀTIE APRĒĶINI: MONTE-KARLO METODE

Algoritms

```
k←0
priekš visiem i no 1 līdz n atkārtot
      x←random vērtība intervālā, kas sakrīt ar kastes pamata garumu
      y←random vērtība intervālā, kas sakrīt ar kastes augstumu
```

ja y<=f(x) tad k←k+1

laukums←(b-a)*h*k/n

Metodes parametri: double a, double b, double h, int n

Atgriežamā vērtība (laukums s): double

Random vērtību intervālā no a līdz b varat dabūt ar šādu paņēmienu:

```
Random r = new Random();
double randomValue = a + (b - a) * r.nextDouble();
```

IZMANTOTIE MATERIĀLI

- Skolotājas Aijas Lūses materiāli
- Formulu attēlojumi no portāla www.uzdevumi.lv