Oplossingen Reeks4 Oef1 – Oef10

Oplossing Oef1

Stel X=4, Y=3

X	Y	Z	U	V
4	3			
		0	4	3
			2	6
			1	12
		12	0	24

Z = X*Y

Stel X=7, Y=4

X	Y	Z	U	V
7	4			
		0	7	4
		4	3	8
		12	1	16
		28	0	32

Z = X*Y

Stel X=0, Y=4

X	Y	Z	U	V
0	4			
		0	0	4

Z = X*Y

Oplossing Oef2

```
{
    // declaratie
    int getal;    // in te lezen getal (groter dan 1)
    int deler;    // loopvariabele voor mogelijke delers
    double max;   // bovengrens voor testen van delers
    int rest;     // rest na deling van getal / deler
```

```
    // inbrengen getal
    System.out.print("Geef een getal: ");
    getal = sc.nextInt();
    // bepalen kleinste positieve deler verschillend van 1
    deler = 2;
    max = Math.sqrt(getal);
    rest = getal % deler;
    while (deler < max && rest != 0)
    {
     deler = deler + 1;
     rest = getal % deler;
    }
    if (rest != 0)
    {
     deler = getal;
    }
    System.out.println("De kleinste positieve deler (≠ 1) is: " + deler);
}
```

Oplossing Oef3 Versie 1

```
{
    // declaratie
    double som;          // som van verschillende termen
    double teller, noemer;// teller en noemer van een term


    // bepalen som via optelling van de termen
    // van links naar rechts
    som = 0;
    // initialisatie teller en noemer op eerste term
    teller = 1;
    for (noemer = 1; noemer <= 1000; noemer = noemer + 1)
    {
     som = som + (teller / noemer);
     teller = -teller;
    }
    System.out.println("De som is: " + som);
}
```

Oplossing Oef3 Versie 2

```
{
     // declaratie
    double som;              // som van verschillende termen
    double teller, noemer;  // teller en noemer van een term


    // bepalen som via optelling van de termen
```

```
    // van links naar rechts
    som = 0;
    // initialisatie teller en noemer op eerste term
    for (noemer = 1, teller = 1; noemer <= 1000;
              teller= -teller, noemer = noemer + 1)
    {
      som = som + (teller / noemer);
    }
    System.out.println("De som is: " + som);
}
```

Oplossing Oef3 Versie 3

```
{
    // declaratie
    double som;          // som van verschillende termen
    double teller, noemer;// teller en noemer van een term


    // bepalen som via optelling van de termen
    // van rechts naar links
    som = 0;
    // initialisatie teller en noemer op eerste term
    for (noemer = 1000, teller = -1; noemer >= 1;
              teller= -teller, noemer = noemer - 1)
    {
      som = som + (teller / noemer);
    }
    System.out.println("De som is: " + som);
}
```

Oplossing Oef3 Versie 4

```
{
    // declaratie
    double somPos;       // som van positieve termen
    double somNeg;       // som van negatieve termen
    double som;          // som van alle termen
    double noemer;       // noemer van een term
```

```
    // bepalen som via aparte optelling van positieve
    // en negatieve termen
    somPos = 0;
    somNeg = 0;
    for (noemer = 1; noemer <= 1000; noemer = noemer + 2)
    {
      somPos = somPos + (1 / noemer);
      somNeg = somNeg - (1 / (noemer + 1));
    }
    som = somPos + somNeg;
    System.out.println("De som is: " + som);
}
```

Oplossing Oef 4.

Algoritme

$$\cos^2(x) = 1 - \frac{2}{2!}x^2 + \frac{2^3}{4!}x^4 - \frac{2^5}{6!}x^6 + \ldots$$

initialisatie

1° term = 1

2° term = $-\frac{2}{2!}x^2 = 1 * \frac{-2}{1*2}x^2$

3° term = $+\frac{2^3}{4!}x^4 = -\frac{2}{2!}x^2 * \frac{-4x^2}{3*4}$

4° term = $-\frac{2^5}{6!}x^6 = +\frac{2^3}{4!}x^4 * \frac{-4x^2}{5*6}$

factor

dubbelnr

Formule: term = term * $\frac{factor}{(dubbelnr-1) * dubbelnr}$

Code oplossing:

```
{
        // declaratie
         double x;          // argument cosinus kwadraat functie
         double term;       // laatst berekende term
         int dubbelnr;      // dubbele van volgnummer van laatst
                            // berekende term
         double factor;     // vast gedeelte -4x2
         double cosKwad;    // cos2(x)

         // invoeren argument
```

```
        System.out.println("Geef argument voor cos: ");
        x = sc.nextDouble();

        // initialisatie op eerste twee termen
        term = -x * x;
        dubbelnr = 4;
        cosKwad = 1 + term;
        factor = -x * x * 4;

        // opeenvolgende nieuwe termen berekenen en optellen
        // bij cosKwad
        while (Math.abs(term) >= 1E-5)
        {
          term = term * factor / ((dubbelnr - 1) * dubbelnr);
          cosKwad = cosKwad + term;
          dubbelnr = dubbelnr + 2;
        }
        System.out.println("De cosinus kwadraat van " + x + " = " + cosKwad);
}
```

Oplossing Oefening 5

```
{
    // declaratie
    int goud, zilver, brons; // grootte sprong medailles
    int spelersnr;       // loopvariabele diverse spelers
    int sprong;          // sprong van bepaalde speler
    final int ATLSPELERS = 12;    // totaal aantal spelers


    // inlezen van alle sprongen en bepalen grootte sprong
    // van gouden,zilveren en bronzen medaille
    goud = 0;
    zilver = 0;
    brons = 0;
    System.out.println("Geef de sprongen in van "+ATLSPELERS+" in: ");
    for (spelersnr = 1; spelersnr <= ATLSPELERS; spelersnr = spelersnr + 1)
    {
       sprong = sc.nextInt();
       if (sprong > goud)
       {
           brons = zilver;
           zilver = goud;
           goud = sprong;
       }
       else if (sprong > zilver)
```

```
            {
              brons = zilver;
              zilver = sprong;
            }
            else if (sprong > brons)
                {
                  brons = sprong;
                }
        }
        System.out.println("De sprong van de gouden medaille is: " + goud);
        System.out.println("De sprong van de zilveren medaille is: " + zilver);
        System.out.println("De sprong van de bronzen medaille is: " + brons);
}
```

Oplossing Oefening 6.

```
{
       // declaratie
        int g1, g2;       // in te voeren strikt positieve getallen
        int h1, h2;       // kopieën van g1 en g2 voor berekening ggd
        int rest;         // rest na deling van h1 / h2



      // inbrengen 2 strikt positieve getallen met inputcontrole
      System.out.print("Geef twee strikt positieve getallen: ");
      g1 = sc.nextInt(); g2 = sc.nextInt();
      while(g1 <= 0 || g2 <= 0)
      {
        System.out.println("Fout beide getallen moeten strikt positief zijn: ");
        System.out.println("Geef beide getallen opnieuw in: ");
        g1 = sc.nextInt(); g2 = sc.nextInt();
      }



      // bepalen ggd van 2 strikt positieve getallen
      // grootste getal = h1, kleinste getal = h2
      if (g1 > g2)
      {
         h1 = g1;
         h2 = g2;
      }
      else
      {
         h1 = g2;
         h2 = g1;
      }
      rest = h1%h2;
```

```
      // bepalen ggd volgens formule uit opgave
      while (rest != 0)
      {
        h1 = h2;
        h2 = rest;
        rest = h1%h2;
      }
      System.out.println("De grootste gemene deler van " +
                         g1 + " en " + g2 + " = " + h2);
}
```

Oplossing Oefening 7

```
{

    // declaratie
      int getal;       // in te voeren strikt positief getal
      int deler;       // loopvariabele voor mogelijke delers
      double max;      // bovengrens voor testen van delers
      int rest;        // rest na deling van getal / deler




      // invoeren getal
      System.out.println("Geef een strikt positief getal in: ");
      getal = sc.nextInt();
      while(getal <= 0)
      {
         System.out.println("Geef een strikt positief getal in: ");
         getal = sc.nextInt();
      }

      // controle priemgetal
      // controleren of er een andere deler bestaat dan 1
      // en het getal zelf
      deler = 2;
      max = Math.sqrt(getal);
      rest = getal % deler;
      while (deler < max && rest != 0)
      {
        deler = deler + 1;
        rest = getal % deler;
      }
      if (rest != 0 && getal > 1)
          System.out.println(getal + " is een priemgetal");
        else
          System.out.println(getal + " is geen priemgetal");
}
```

Oplossing oefening 8

```
{
    // declaratie
    int getal;   // in te voeren strikt positief getal
    int term1, term2, term3;// 3 opeenvolgende termen in de rij v. fibonacci

    // invoeren getal
    getal = sc.nextInt();

    // controle fibonacci getal
    // initialiseer eerste twee termen
    term1 = 0;
    term2 = 1;
    while (getal > term2)
    {
      term3 = term1 + term2;
      term1 = term2;
      term2 = term3;
    }
    if (term2 == getal)
      System.out.println(getal + " is een fibonacci-getal");
    else
      System.out.println(getal + " is geen fibonacci-getal");
}
```

Oplossing Oefening 9

```
{
    // declaratie
    int g1, g2, g3;   // 3 opeenvolg. in te voeren getallen
    int atlLokMax;    // aantal lokale maxima

    // invoeren eerste 3 getallen
    g1 = sc.nextInt();
    g2 = sc.nextInt();
    g3 = sc.nextInt();
    atlLokMax = 0;
    while (g3 != -1)
    {
      if (g2 > g1 && g2 > g3)
      {
        System.out.println(g1+" "+g2+" "+g3);
        atlLokMax = atlLokMax + 1;
      }
      g1 = g2;
      g2 = g3;
      g3 = sc.nextInt();
    }
```

```
    System.out.println("Aantal lokale maxima: " + atlLokMax);
}
```

Oplossing Oefening 9 versie 2

```
{
       // declaratie
    int g1, g2, g3;   // 3 opeenvolg. in te voeren getallen
    int atlLokMax;    // aantal lokale maxima

    // invoeren eerste 3 getallen
    atlLokMax = 0;
    g1 = sc.nextInt();
    if (g1 != -1)
    {
     g2 = sc.nextInt();
     if (g2 != -1)
     {
      g3 = sc.nextInt();
      while (g3 != -1)
      {
       if (g2 > g1 && g2 > g3)
       {
        System.out.println(g1+" "+g2+" "+g3);
        atlLokMax = atlLokMax + 1;
       }
       g1 = g2;
       g2 = g3;
       g3 = sc.nextInt();
      }
     }
    }
    System.out.println("Aantal lokale maxima: " + atlLokMax);
}
```

Oplossing Oefening 10

```
{
      // declaratie
       int som;          // som van alle getallen
       int getal;        // laatst ingelezen getal
       int vorig;        // vorig ingelezen getal
       int lenLanStij;   // lengte langst stijgende reeks
       int somLanStij;   // som langst stijgende reeks
```

```
int lenLanDal;    // lengte langst dalende reeks
int somLanDal;    // som langst dalende reeks
int lenHuiStij;   // lengte huidig stijgende reeks
int somHuiStij;   // som huidig stijgende reeks
int lenHuiDal;    // lengte huidig dalende reeks
int somHuiDal;    // som huidig dalende reeks

// invoeren eerste getal
getal = sc.nextInt();

// initialisatie van alle reeksen op 0
som = 0;
lenLanStij = 0;
somLanStij = 0;
lenLanDal = 0;
somLanDal = 0;
lenHuiStij = 0;
somHuiStij = 0;
lenHuiDal = 0;
somHuiDal = 0;

// initialisatie vorig op fictief getal om te starten
// met stijgende reeks
vorig = getal - 1;
// volledige reeks getallen overlopen
while (getal != 0)
{
 if (getal > vorig)
 { // Verwerk stijgende reeks
   // als we met een stijging zitten
   // dan zolang de stijging blijft duren, en niet aan einde reeks
   while (getal > vorig && getal != 0)
   {
    // Pas huidig stijgende reeks aan
    lenHuiStij = lenHuiStij + 1;
    somHuiStij = somHuiStij + getal;
    som = som + getal;
    vorig = getal;
    getal = sc.nextInt();
   }
   // na de lus begint de reeks weer te dalen (of is de reeks gedaan)
   // nu kunnen we kijken of de reeks van stijgende getallen groter is dan
   // de reeks die we tot nu toe hadden
   // Pas eventueel langst stijgend reeks aan
   if (lenHuiStij > lenLanStij)
   {
     // de rij is langer
     lenLanStij = lenHuiStij;
     somLanStij = somHuiStij;
   }
```

```
      else
      {
        // de rij is even lang, maar de som van de rij is groter
        if (lenHuiStij == lenLanStij&& somHuiStij > somLanStij)
          somLanStij = somHuiStij;
      }
      // Initialiseer huidig dalende reeks op vorig getal
       lenHuiDal = 1;
       somHuiDal = vorig;
    }
    else
    { // Verwerk dalende reeks
      // als we met een daling zitten
      // dan zolang de daling blijft duren,
      // en niet aan einde reeks
      while (getal < vorig && getal != 0)
      {
        // Pas huidig dalende reeks aan
        lenHuiDal = lenHuiDal + 1;
        somHuiDal = somHuiDal + getal;
        som = som + getal;
        vorig = getal;
        getal = sc.nextInt();
      }
      // na de lus begint de reeks weer te stijgen (of is de reeks gedaan)
      // nu kunnen we kijken of de reeks van dalende getallen groter is dan
      // de reeks die we tot nu toe hadden
      // Pas eventueel langst dalende reeks aan
      if (lenHuiDal > lenLanDal)
      {
        // de rij is langer
        lenLanDal = lenHuiDal;
        somLanDal = somHuiDal;
      }
      else
      {
      // de rij is even lang, maar de som van de rij is groter
        if (lenHuiDal == lenLanDal && somHuiDal >somLanDal)
          somLanDal = somHuiDal;
      }
      // Initialiseer huidig stijgende reeks op vorig getal
      lenHuiStij = 1;
      somHuiStij = vorig;
    }
  }
  // Resultaat
   System.out.println("De som van alle elementen = " + som);
   System.out.println("De lengte van de langst stijgende reeks = " +
                           lenLanStij);
   System.out.println("De som van de langst stijgende reeks = "+
                           somLanStij);
```

```
        System.out.println("De lengte van de langst dalende reeks = "+
                                    lenLanDal);
        System.out.println("De som van de langst dalende reeks = " +
                                   somLanDal);
}
```