Cvičenie 3 - Podmienený príkaz a cyklus

Podmienený príkaz

Základný príkaz má rovnako ako v ostatných programovacích jazykoch blokovú štruktúru. Jednotlivé bloky sú oddelené čiarkou.

V najjednoduchšej verzii má príkaz If štruktúru tvorenú dvomi blokmi. Bloky sú v tomto prípade: testovacia podmienka, blok True.

```
If [ podmienka,
    čo robiť ak je podmienka splnená]
```

V základnej verzii má príkaz If štruktúru tvorenú tromi blokmi. Bloky sú v tomto prípade: testovacia podmienka, blok True, blok False

```
If [ podmienka,
    čo robiť ak podmienka je splnená,
    čo robiť ak podmienka nie je splnená]
```

Mathematica na rozdiel od ostatných programovacích jazykov, umožňuje aj tzv. rozšírenú syntax podmieneného príkazu If - teda programovú štruktúru, ktorá je tvorená štyrmi blokmi.

```
If [ podmienka,
    čo robiť ak podmienka je splnená,
    čo robiť ak podmienka nie je splnená,
    čo robiť ako nedokážem podmienku vyhodnotiť]
```

Ak nevie rozhodnúť, či je podmienka splnená, tak poskytne ako výsledok tretí blok.

Príklad

Zostavte program, ktorý porovná dve čísla a, b. Čísla a a b sú zadané programátorom na začiatku programu. Ak je číslo $a < b$ na obrazovku sa vypíše správa "číslo a je menšie ako číslo b".
Funkčnosť programu otestujte tak, že overíte funkčnosť všetkých jeho vetiev.

In[ ]:=
```
Clear[a, b]
a = 2;
b = 5;
If[a < b,
 Print["Číslo ", a, " je menšie ako číslo ", b]
  ]
```
```
Číslo 2 je menšie ako číslo 5
```

Príklad

Zostavte program, ktorý porovná dve čísla a, b. Čísla *a* a *b* sú zadané programátorom na začiatku programu. Ak je číslo $a < b$ na obrazovku sa vypíše správa "číslo *a* je menšie ako číslo *b*". V opačnom prípade sa na obrazovku vypíše správa "číslo *b* je menšie ako číslo *a*".
Funkčnosť programu otestujte tak, že overíte funkčnosť všetkých jeho vetiev.

In[ ]:=
```
Clear[a, b]
a = 2;
b = 5;
If[a < b,
 Print["Číslo ", a, " je menšie ako číslo ", b],
 Print["Číslo ", b, " je menšie ako číslo ", a]
  ]
```
```
Číslo 2 je menšie ako číslo 5
```

In[ ]:=
```
Clear[a, b]
a = 12;
b = 5;
If[a < b,
 Print["Číslo ", a, " je menšie ako číslo ", b],
 Print["Číslo ", b, " je menšie ako číslo ", a]
  ]
```
```
Číslo 5 je menšie ako číslo 12
```

Príklad

Zostavte program, ktorý porovná dve čísla a, b. Čísla *a* a *b* sú zadané programátorom na začiatku programu. Ak je číslo $a < b$ na obrazovku sa vypíše správa "číslo *a* je menšie ako číslo *b*". V opačnom prípade sa na obrazovku vypíše správa "číslo *b* je menšie ako číslo *a*". V prípade, že program nevie rozhodnúť, ktoré z týchto dvoch tvrdení je pravdivé, vypíše o tom správu na obrazovku.
Funkčnosť programu otestujte tak, že overíte funkčnosť všetkých jeho vetiev.

In[ ]:=
```
Clear[a, b]
a = 2;
b = 5;
If[a < b,
 Print["Číslo ", a, " je menšie ako číslo ", b],
 Print["Číslo ", b, " je menšie ako číslo ", a],
 Print["Neviem rozhodnúť o pravdivosti tvrdenia"]
  ]
```

```
Číslo 2 je menšie ako číslo 5
```

In[ ]:=
```
Clear[a, b]
a = 12;
b = 5;
If[a < b,
 Print["Číslo ", a, " je menšie ako číslo ", b],
 Print["Číslo ", b, " je menšie ako číslo ", a],
 Print["Neviem rozhodnúť o pravdivosti tvrdenia"]
  ]
```

```
Číslo 5 je menšie ako číslo 12
```

In[ ]:=
```
Clear[a, b]
a = 12 + 2 I;
b = 5;
If[a < b,
 Print["Číslo ", a, " je menšie ako číslo ", b],
 Print["Číslo ", b, " je menšie ako číslo ", a],
 Print["Neviem rozhodnúť o pravdivosti tvrdenia"]
  ]
```

Less: Invalid comparison with 12 + 2 *i* attempted.

```
Neviem rozhodnúť o pravdivosti tvrdenia
```

Príklad

Zostavte program, ktorý testuje, či ľubovolné reálne číslo *c* patrí do intervalu (*a*, *b*), kde *a*, *b* sú ľubovolné reálne čísla také, že interval (*a*, *b*) má zmysel. Čísla *a*, *b* a *c* sú zadané programátorom na začiatku programu. Ak je číslo $c \in (a, b)$ na obrazovku sa vypíše správa "číslo *c* je z intervalu (*a*, *b*)". V opačnom prípade sa vypíše správa "číslo *c* nie je z intervalu (*a*, *b*)".
Funkčnosť programu otestujte tak, že overíte funkčnosť všetkých jeho vetiev.

In[ ]:=
```
Clear[a, b, c]
a = 4;
b = 6;
c = 5;
If[a < c < b,
     Print["číslo ", c, " je z intervalu (", a, ",", b, ")"],
     Print["číslo ", c, " nie je z intervalu (", a, ",", b, ")"]
    ]
```
```
číslo 5 je z intervalu (4,6)
```

In[ ]:=
```
Clear[a, b, c]
a = 4;
b = 6;
c = 15;
If[a < c < b,
     Print["číslo ", c, " je z intervalu (", a, ",", b, ")"],
     Print["číslo ", c, " nie je z intervalu (", a, ",", b, ")"]
    ]
```
```
číslo 15 nie je z intervalu (4,6)
```

Príklad

Zostavte program, ktorý testuje, či ľubovolné reálne číslo c patrí do intervalu (a, b), kde a, b sú ľubovolné reálne čísla také, že interval (a, b) má zmysel. Čísla a, b a c sú zadané programátorom na začiatku programu. Ak je číslo $c \in (a, b)$ na obrazovku sa vypíše správa "číslo c je z intervalu (a, b)". V opačnom prípade sa vypíše správa "číslo c nie je z intervalu (a, b)".
Program bude zároveň testovať, či je interval správne zadaný.
Funkčnosť programu otestujte tak, že overíte funkčnosť všetkých jeho vetiev.

In[ ]:=
```
Clear[a, b, c]
a = 7;
b = 6;
c = 5;
If[a < b,

 Print["Interval je zadaný správne."];
 If[a < c  < b,
      Print[c, " je z intervalu (", a, ",", b, ")"],
      Print[c, " nie je z intervalu (", a, ",", b, ")"]
      ],

 Print["b<a, Interval nie je zadaný správne."]

   ]
```

```
b<a, Interval nie je zadaný správne.
```

In[ ]:=
```
Clear[a, b, c]
a = 3;
b = 6;
c = 5;
If[a < b,

 Print["Interval je zadaný správne."];
 If[a < c  < b,
      Print[c, " je z intervalu (", a, ",", b, ")"],
      Print[c, " nie je z intervalu (", a, ",", b, ")"]
      ],

 Print["b<a, Interval nie je zadaný správne."]

   ]
```

```
Interval je zadaný správne.
5 je z intervalu (3,6)
```

Príklad na samostatné počítanie

Zostavte program, ktorý vypočíta riešenie lineárnej rovnice $a\,x + b = 0$, kde a, b sú ľubovolné reálne čísla. Program musí otestovať, či $a \neq 0$. Čísla a a b sú zadané programátorom na začiatku programu. Program vypíše správy "Riešenie rovnice je ...." . V prípade chybného zadania rovnice vypíše správu "Rovnica je chybne zadaná, delíš nulou."
Funkčnosť programu otestujte tak, že overíte funkčnosť všetkých jeho vetiev.

Príklad na samostatné počítanie

Zostavte program, ktorý zistí, či hodnota x ∈ <a, b>. Čísla a, b a x sú zadané programátorom na začiatku programu. Program otestuje, či reálne číslo x patrí do intervalu (a, b), kde a, b sú ľubovolné reálne čísla také, že interval (a, b) má zmysel. Ak áno, vypočíta odmocninu z tohto čísla, ak nie, nájde jeho tretiu mocninu. Výsledok zapíše na obrazovku.

Príklad na samostatné počítanie

Napíšte program, ktorý vypíše absolútnu hodnotu čísla, ktoré zadal užívateľ.
Absolútna hodnota čísla $|a|$ je definovaná nasledovne:

- Ak $a \geq 0$ tak $|a| = a$.
- Ak $a < 0$ tak $|a| = -a$.

Návod na riešenie:
Vstup zadaný z klávesnice môžeme získať pomocou príkazu Input [ ].

```
In[ ]:= a = Input["Zadaj číslo a"]
```

```
Out[ ]= 5
```

Príklad na samostatné počítanie

Napíšte program, ktorý vypíše či rok (napr. 2004), ktorý zadal užívateľ je z aktuálneho storočia, z minulého storočia, alebo z ďalekej minulosti, alebo z ďalekej budúcnosti.

Príklad na samostatné počítanie

Napíšte program pomocou podmienene definovanej funkcie, ktorý vypíše či rok (napr. 2004), ktorý zadal užívateľ je z aktuálneho storočia, z minulého storočia, alebo z ďalekej minulosti, alebo z ďalekej budúcnosti.

Príklad na samostatné počítanie

Napíšte program, ktorý vypíše či rok (napr. 2004), ktorý zadal užívateľ je z aktuálneho storočia, z minulého storočia, alebo z ďalekej minulosti.

Príklad na samostatné počítanie

Napíšte program, ktorý vypíše či rok (napr. 2004), ktorý zadal užívateľ je priestupný.

Príklad na samostatné počítanie

Kvadratická rovnica $a\,x^2 + b\,x + c = 0$ má v reálnych čísla tri prípady riešenia, čo zistíme výpočtom determinantu $D = b^2 - 4\,a\,c$.

Potom platí
- Ak je diskriminant D kladný, rovnica má dva reálne korene.
- Ak je diskriminant D rovný nule, rovnica má jeden reálny dvojnásobný koreň.

- Ak je diskriminant D záporný, rovnica má dva komplexné korene.

Napíšte program, ktorý túto situáciu pre premenné zadané užívateľom odsimuluje.

Príklad na samostatné počítanie

Napíšte program, ktorý pre zadané hodnory odporov R_1 a R_2 vypočíta výsledný odpor podľa toho, ako sú zapojené (sériovo, alebo paralelne). Program komunikuje s užívateľom, vyžiada si zadanie údajov a spôsobu zapojenia od užívateľa.

Príklad na samostatné počítanie

Napíšte program, ktorý po zadaní dátumu narodenia (deň, mesiac, rok) užívateľom vypíše znamenie horoskopu v ktorom sa užívateľ narodil.

Príklad na samostatné počítanie

Napíšte program, ktorý načíta 3 čísla a vypíše najväčšie z nich. Vymyslite aspoň tri principiálne rôzne spôsoby ako to naprogramovať.

Príklad na samostatné počítanie

Napíšte program, ktorý po zadaní dvoch čísel zistí, či ich súčin je rovný nule alebo nie. Vymyslite aspoň tri principiálne rôzne spôsoby ako to naprogramovať.

Príklad na samostatné počítanie

Napíšte program, ktorý po zadaní mesiaca vypíše ročné obdobie do ktorého patrí. Vymyslite aspoň dva principiálne rôzne spôsoby ako to naprogramovať.

Príklad na samostatné počítanie

Napíšte program - kalkulačka, ktorý po zadaní dvoch čísel a operácie, ktorú chceme vykonať nám výpočet zrealizuje a zapíše na obrazovku výsledok.
Operácie budeme zadávať pomocou znakov "+", "-". "*", "/". Vymyslite aspoň dva principiálne rôzne spôsoby ako to naprogramovať.

Príklad na samostatné počítanie

Napíšte program, ktorý zistí, či číslo zadané užívateľom je párne alebo nepárne. Vymyslite aspoň tri principiálne rôzne spôsoby ako to naprogramovať.

Príklad na samostatné počítanie

Napíšte program, ktorý vypočíta objem a obsah kvádra podľa toho, čo si užívateľ želá vypočítať. Program s uťívateľom bude komunikovať. Vymyslite aspoň tri principiálne rôzne spôsoby ako to naprogramovať.

Príklad na samostatné počítanie

Náhodne vygenerujte celé číslo z množiny {1, 2, ..., 10}. Ak je vygenerované číslo 1 alebo 2, vypíšte na obrazovku správu: ”Pravdepodobnosť je 20%”, v prípade vygenerovanie 3 alebo 4 vypíšte: ”Pravdepodobnosť je 30%”. V ostatných prípadoch vypíšte: ”Pravdepodobnosť je 50%”.

Cyklus - Do, s pevným aj s variabilným počtom opakovaní

Základná štruktúra príkazu cyklu má tvar:

```
Do[blok príkazov, {List}]
```

alebo

```
Do[
    príkaz1;
    príkaz2;
    príkaz3,   {List}]
```

Blok príkazov rovnako ako v prípade podmieneného príkazu If môže byť tvorený jedným príkazom, alebo niekoľkými príkazmi oddelených bodkočiarkami. Za posledným príkazom v bloku bude nasledovať čiarka, ktorá hovorí, že teraz je koniec vykonávacej časti cyklu a nasleduje časť, v ktorej hovoríme koľko krát a ako sa má cyklus zopakovať. Táto časť môže mať tvar

- {n} - krát sa zopakujú príkazy v tele cyklu,
- {i, n} - riadiaca premenná i nadobúda hodnoty od 1 do a podľa toho sa aj cyklus opakuje,
- {i, štart, koniec} - riadiaca premenná i nadobúda hodnoty od štart hodnoty do koniec hodnoty a podľa toho sa aj cyklus opakuje,
- {i, štart, koniec, krok} - riadiaca premenná i nadobúda hodnoty od štartovacej hodnoty do koniec hodnoty s krokom krok a podľa toho sa aj cyklus opakuje

Príklad

Pomocou cyklu vygeneruj 4 náhodné reálne čísla z intervalu [0, 1].
Pomocou cyklu vygeneruj 4 náhodné reálne čísla z intervalu [5, 10].
Pomocou cyklu vygeneruj 4 náhodné celé čísla z intervalu [–2, 10].

In[ ]:=
```
Do[Print[Random[]], {4}]
```

```
0.107172
0.524977
0.0124017
0.590817
```

```
In[ ]:= Do[Print[Random[Real, {5, 10}]], {4}]
```

```
7.22854
5.86975
9.93225
8.91452
```

```
In[ ]:= Do[Print[Random[Integer, {-2, 10}]], {4}]
```

```
10
5
-2
-2
```

Príklad

Vytvorte množinu *A* tvorenú 12-timi náhodných reálnych čísel z intervalu [1, 10]. Zistite

1. počet čísel z danej množiny, ktorých hodnota je menšia ako 5,
2. súčet čísel z danej množiny, ktorých hodnota je menšia ako 5 .
3. počet a súčet čísel z danej množiny, ktorých hodnota je menšia ako 5 .

Vytvoríme množinu *A* s požadovanými vlastnosťami. Ukážeme si, ako sa dajú z množiny A vybrať jednotlivé prvky.

```
In[ ]:= A = Table[Random[Real, {1, 10}], {i, 1, 12}]
```

```
Out[ ]= {1.61683, 2.8867, 7.37512, 3.10697, 5.85422,
  1.03664, 8.1682, 4.17648, 4.16404, 9.4997, 7.95364, 6.22863}
```

Vyberieme prvý prvok, piaty prvok a deviaty (štvrtý od konca) prvok

```
In[ ]:= A[[1]]
```

```
Out[ ]= 1.61683
```

```
In[ ]:= A[[5]]
```

```
Out[ ]= 5.85422
```

```
In[ ]:= A[[-4]]
```

```
Out[ ]= 4.16404
```

1. Zistite počet čísel z danej množiny, ktorých hodnota je menšia ako 5.

Nápad 1

In[ ]:=
```
Clear[pocet]
pocet = 0;
Do[
     If[A[[i]] < 5, pocet = pocet + 1],
 {i, 1, 12}]
Print["Pocet cisel mensich ako 5 je ", pocet]
```

```
Pocet cisel mensich ako 5 je 6
```

Nápad 2 - používame konvenciu z programovacieho jazyka C

In[ ]:=
```
Clear[pocet]
pocet = 0;
Do[
     If[A[[i]] < 5, pocet++],
 {i, 1, 12}]
Print["Pocet cisel mensich ako 5 je ", pocet]
```

```
Pocet cisel mensich ako 5 je 6
```

2. Zistite súčet čísel z danej množiny, ktorých hodnota je menšia ako 5.

Nápad 1

In[ ]:=
```
Clear[sucet]
sucet = 0;
Do[
     If[A[[i]] < 5, sucet = sucet + A[[i]]], {i, 1, 12}]
Print["Sucet cisel mensich ako 5 je ", sucet]
```

```
Sucet cisel mensich ako 5 je 16.9877
```

Nápad 2 - používame konvenciu z programovacieho jazyka C

In[ ]:=
```
Clear[sucet]
sucet = 0;
Do[
     If[A[[i]] < 5, sucet += A[[i]]], {i, 1, 12}]
Print["Sucet cisel mensich ako 5 je ", sucet]
```

```
Sucet cisel mensich ako 5 je 16.9877
```

3. počet a súčet čísel z danej množiny, ktorých hodnota je menšia ako 5.

Nápad 1

In[ ]:=
```
Clear[pocet]
pocet = 0;
sucet = 0;
Do[
     If[A[[i]] < 5, pocet = pocet + 1;
                    sucet = sucet + A[[i]]
       ],
 {i, 1, 12}]
Print["Pocet cisel mensich ako 5 je ", pocet]
Print["Sucet cisel mensich ako 5 je ", sucet]
```

```
Pocet cisel mensich ako 5 je 6

Sucet cisel mensich ako 5 je 16.9877
```

Nápad 2 - používame konvenciu z programovacieho jazyka C

In[ ]:=
```
Clear[pocet]
pocet = 0;
sucet = 0;
Do[
     If[A[[i]] < 5, pocet++;
                    sucet += A[[i]]
       ],
 {i, 1, 12}]
Print["Pocet cisel mensich ako 5 je ", pocet]
Print["Sucet cisel mensich ako 5 je ", sucet]
```

```
Pocet cisel mensich ako 5 je 6

Sucet cisel mensich ako 5 je 16.9877
```

Príklad

Vypočítajte druhú odmocninu z nezáporného čísla a pomocou Newtonovho algoritmu. Použite iteračný predpis $x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$

Konkrétne napríklad $\{x_{n+1}\}$ vypočítaná podľa vzťahov $x_{n+1} = 1/2\left(x_n + \frac{2}{x_n}\right)$ bude konvergovať (blížiť sa) ku číslu $\sqrt{2}$.

Konkrétne napríklad $\{x_{n+1}\}$ vypočítaná podľa vzťahov $x_{n+1} = 1/2\left(x_n + \frac{5}{x_n}\right)$ bude konvergovať (blížiť sa) ku číslu $\sqrt{5}$.

Nápad 1 - použijeme základné procedurálne programovanie

Výpočet necháme zopakovať pevný počet opakovaní - v tomto príklade 7 x

```
In[ ]:= Clear[x]
        a = 2;
        x = 1;
        Do[
            x = (x + a / x) / 2 // N;
            Print[x],
            {n, 1, 7}
          ]
```

```
1.5

1.41667

1.41422

1.41421

1.41421

1.41421

1.41421
```

Nápad 2 - použijeme indexovanú premennú

Tento spôsob je efektívnejší, pretože nám umožňuje následne s jednotlivými hodnotami x_n manipulovať (vypísať, vykresliť...)

```
In[ ]:= Clear[x, a]
        a = 2;
        x[1] = 1;
        Do[
            x[n + 1] = 1/2 (x[n] + a/x[n]) // N,
            {n, 1, 7}
            ]
        Table[x[n], {n, 1, 7}]
```

```
Out[ ]= {1, 1.5, 1.41667, 1.41422, 1.41421, 1.41421, 1.41421}
```

Nápad 2A - použijeme indexovanú premennú a počet opakovaní necháme voliteľný

Tento spôsob je efektívnejší, pretože nám umožňuje následne s jednotlivými hodnotami x_n manipulovať (vypísať, vykresliť...)

In[ ]:=
```
Clear[x, a]
a = 2;
x[1] = 1;
pocetopakovani = 9;
Do[
    x[n + 1] = 1/2 (x[n] + a/x[n]) // N,
    {n, 1, pocetopakovani}
   ]
Table[x[n], {n, 1, pocetopakovani}]
```

Out[ ]= {1, 1.5, 1.41667, 1.41422, 1.41421, 1.41421, 1.41421, 1.41421, 1.41421}

Nápad 3 - použijeme indexovanú premennú, pridáme aj testovaciu podmienku

Tento spôsob je efektívnejší, pretože nám umožňuje následne s jednotlivými hodnotami x_n manipulovať (vypísať, vykresliť...)
Do výpočtu pridáme "zastavovaciu podmienku". Zastavovacia podmienka je podmienka, ktorá hovorí "zastav výpočet vtedy ak ..." , napríklad, ak sa hodnota dvoch po sebe nasledujúcich vypočítaných výsledkov "veľmi málo líši." - napríklad $\left| x_{n+1} - x_n \right| < 10^{-6}$

In[ ]:=
```
Clear[x, a]
a = 2;
x[1] = 1;
pocetopakovani = 9;
Do[
    x[n + 1] = 1/2 (x[n] + a/x[n]) // N;
     Print[x[n + 1]];
    If [Abs[x[n - 1] - x[n]] < 10^(-6), Break[]],
    {n, 1, pocetopakovani}
   ]
```

```
1.5
1.41667
1.41422
1.41421
1.41421
1.41421
```

Nápad 4 -
použijeme indexovanú premennú, pridáme aj testovaciu podmienku,

hodnotu riadiacej premennej v cykle uchováme

Tento spôsob je efektívnejší, pretože nám umožňuje následne s jednotlivými hodnotami x_n manipulovať (vypísať, vykresliť...)
Do výpočtu pridáme "zastavovaciu podmienku". Zastavovacia podmienka je podmienka, ktorá hovorí "zastav výpočet vtedy ak ..." , napríklad, ak sa hodnota dvoch po sebe nasledujúcich vypočítaných výsledkov "veľmi málo líši." - napríklad $\left| x_{n+1} - x_n \right| < 10^{-6}$
V okamžiku zastavenia výpočtu v cykle hodnotu riadiacej premennj uchováme a následne použijeme na vypísanie hodnôt do tabuľky

In[ ]:=
```
Clear[x, a]
a = 2;
x[1] = 1;
pocetopakovani = 20;
skutocnypocetopakovani = Catch[
   Do[
      x[n + 1] = 1/2 (x[n] + a/x[n]) // N;
      If [Abs[x[n - 1] - x[n]] < 10^(-6), Throw[n]; Break[]],
     {n, 1, pocetopakovani}
     ]
  ];
Table[x[n], {n, 1, skutocnypocetopakovani}]
```

Out[ ]=
```
{1, 1.5, 1.41667, 1.41422, 1.41421, 1.41421}
```

In[ ]:=
```
TableForm[Table[{n, x[n]}, {n, 1, skutocnypocetopakovani}],
     TableHeadings → {None, {"n", "x_n"}},
     TableSpacing → {1, 4}]
```

Out[ ]//TableForm=

n	"x_n"
1	1
2	1.5
3	1.41667
4	1.41422
5	1.41421
6	1.41421

Príklad

Lodné motory so vstrekovaním Common-Rail majú menovitý výkon P_m v rozsahu 12 000 až 70 000 kW. Počas plavby motor pracuje na výkon P rovný 75 % menovitého výkonu. Špecifická spotreba paliva je $m_p = 170\ g/\text{kWh}$. Zostavte program, ktorý vypíše na monitor tabuľku vyjadrujúcu závislosť spotreby paliva S_p [kg/h] od výkonu motora P . Počítajte s krokom $\Delta P = 4350$ kW.

NÁVOD :

Ak dosadíme špecifickú spotrebu v g/kWh a výkon v kW , spotrebu paliva v kg/h vypočítame podľa vzorca $S_p = \frac{m_p P}{1000}$. Hranice pre výkon P vypočítame zo vzťahu $P = 0,75 . P_m$

Nápad 1 - najzákladnejšie procedurálne programovanie

In[ ]:=
```
mp = 170;
Pmin = 12 000;
Pmax = 70 000;
krok = 4350;

Print["P[kW]     Sp[kg/h]"]
Print["___________________"]
Do[   P = 0.75 * Pm;
      Sp = mp * P / 1000;
      Print[P, "        ", Sp],
   {Pm, Pmin, Pmax, krok}
  ]
```

```
P[kW]     Sp[kg/h]

___________________
9000.        1530.
12 262.5       2084.63
15 525.       2639.25
18 787.5       3193.88
22 050.       3748.5
25 312.5       4303.13
28 575.       4857.75
31 837.5       5412.38
35 100.       5967.
38 362.5       6521.63
41 625.       7076.25
44 887.5       7630.88
48 150.       8185.5
51 412.5       8740.13
```

Nápad 2 - naprogramované pomocou definovania funkcie

In[ ]:=
```
Clear[mp, Pmin, Pmax, krok, Sp]
mp = 170;
Pmin = 12 000;
Pmax = 70 000;
krok = 4350;

Sp[P_] := (mp * 0.75 * P / 1000);
vysledok = Table[{0.75 * P, Sp[P]}, {P, Pmin, Pmax, krok}]
TableForm[vysledok,
 TableHeadings → {None, {"P [kW]", "Sp [kg/h]"}}, TableSpacing → {1, 4}]
```

Out[ ]=
```
{{9000., 1530.}, {12 262.5, 2084.63}, {15 525., 2639.25},
 {18 787.5, 3193.88}, {22 050., 3748.5}, {25 312.5, 4303.13},
 {28 575., 4857.75}, {31 837.5, 5412.38}, {35 100., 5967.}, {38 362.5, 6521.63},
 {41 625., 7076.25}, {44 887.5, 7630.88}, {48 150., 8185.5}, {51 412.5, 8740.13}}
```

Out[ ]//TableForm=

P [kW]	"Sp [kg/h]"
9000.	1530.
12 262.5	2084.63
15 525.	2639.25
18 787.5	3193.88
22 050.	3748.5
25 312.5	4303.13
28 575.	4857.75
31 837.5	5412.38
35 100.	5967.
38 362.5	6521.63
41 625.	7076.25
44 887.5	7630.88
48 150.	8185.5
51 412.5	8740.13

Nápad 2A - naprogramované pomocou definovania funkcie

V úlohe je zbytočné definovať samostatne premenné Pmin, Pmax a krok . Stačí potrebné hodnoty dosadiť priamo do výpočtu

```
In[ ]:= Clear[Sp]

        Sp[P_] := 170 * 0.75 * P / 1000;
        vysledok = Table[{0.75 * P, Sp[P]}, {P, 12000, 70000, 4350}]
        TableForm[vysledok,
         TableHeadings → {None, {"P [kW]", "Sp [kg/h]"}}, TableSpacing → {1, 4}]

Out[ ]= {{9000., 1530.}, {12262.5, 2084.63}, {15525., 2639.25},
         {18787.5, 3193.88}, {22050., 3748.5}, {25312.5, 4303.13},
         {28575., 4857.75}, {31837.5, 5412.38}, {35100., 5967.}, {38362.5, 6521.63},
         {41625., 7076.25}, {44887.5, 7630.88}, {48150., 8185.5}, {51412.5, 8740.13}}
```

Out[ ]//TableForm=

P [kW]	"Sp [kg/h]"
9000.	1530.
12262.5	2084.63
15525.	2639.25
18787.5	3193.88
22050.	3748.5
25312.5	4303.13
28575.	4857.75
31837.5	5412.38
35100.	5967.
38362.5	6521.63
41625.	7076.25
44887.5	7630.88
48150.	8185.5
51412.5	8740.13

Príklad

Na rezanie závitových dier do kovových a plastických materiálov sa používajú závitníky. Veľkosť podbrúsenia závitníka h sa vypočíta podľa vzťahu $h = \frac{\pi . d . \tan \alpha}{n}$, kde d je priemer závitníka v mm, n je počet drážok, α je uhol chrbta.

Zostavte program, ktorý vypíše tabuľku veľkosti podbrúsenia v závislosti od priemeru závitníka pri n = 3, $\alpha = 3°$ (pozor Mathematica počíta v radiánoch). Počítajte pre $d = d_{min}$ až d_{max} [mm], kde $d_{min} = 2$, $d_{max} = 24$ mm.

Návod : α[°] = $\alpha * \pi$ /180 [rad].

Nápad 1 - najzákladnejšie procedurálne programovanie

In[ ]:=
```
Clear[n, α, h, dmax, d, dmin, dmax]
dmin = 2;
dmax = 24;
n = 3;
α = 3 Degree;

Print["d        h"];
Print["__________________"];
Do[ h = (Pi * d * Tan[α]) / n;
    Print[d, "    ", h // N],
      {d, dmin, dmax, 4}
    ]
```

```
d        h

__________________

2    0.109763

6    0.329288

10    0.548813

14    0.768338

18    0.987863

22    1.20739
```

Nápad 2 - naprogramované pomocou definovania funkcie

In[ ]:=
```
Clear[n, α, h, dmax, d, dmin, dmax]
dmin = 2;
dmax = 24;
n = 3;
α = 3 Degree;

h[d_] = (Pi * d * Tan[α]) / n;

vysledok = Table[{d, h[d]}, {d, dmin, dmax, 4}];
TableForm[vysledok // N, TableHeadings → {None, {"d[mm]", "h [mm]"}},
          TableSpacing → {1, 4}]
```

Out[ ]//TableForm=

d[mm]	h [mm]
2.	0.109763
6.	0.329288
10.	0.548813
14.	0.768338
18.	0.987863
22.	1.20739

Príklad

Hriadeľ uložený v ložiskách so m.g vzdialenosťou l zaťažený v strede zotrvačníkom s hmotnosťou m prenáša výkon P elektromotora s otáčkami n.

Pre minimálny priemer hriadeľa d platí $d = \sqrt[3]{10\, M_{or} / \sigma_{dov}}$

kde $M_{or} = \sqrt{M_o^2 + 0.75\, M_k^2}$ je redukovaný ohybový moment,

M_o = m.g.l/4 je ohybový moment,

M_k= P/(2 π. n) je krútiaci moment,

σ_{dov} je dovolené napätie.

Zostavte program, ktorý vypíše tabuľku minimálnych priemerov hriadeľa v závislosti od vzdialenosti l. Počítajte s hodnotami m = 300 kg, n = 50 s^{-1}, P = 5.5 kW = 5.5 × 10^3 W, σ_{dov} = 107 MPa = 1.07 × 10^8 Pa (dosadzujte v základných jednotkách).

Nápad 1 - najzákladnejšie procedurálne programovanie

In[ ]:=
```
Clear[m, n, P, σ, g, lmin, lmax, krok, Mk, Mo, Mor, d]
m = 300;
n = 50;
P = 5.5 * 10^3;
σ = 1.07 * 10^8;
g = 9.81;

lmin = 0.4;
lmax = 0.8;
krok = 0.05;

Print["l[m]      d[m]"];
Print["___________________"];
Do[Mk = P / (2 Pi * n);
    Mo = m * g * l / 4;
    Mor = Sqrt[Mo^2 + 0.75 Mk^2];
    d = (10 Mor / σ) ^ (1 / 3);
    Print[l, "      ", d],
    {l, lmin, lmax, krok}
   ];
```

l[m]	d[m]
0.4	0.0301991
0.45	0.0314054
0.5	0.0325258
0.55	0.0335741
0.6	0.0345609
0.65	0.0354944
0.7	0.0363812
0.75	0.0372269
0.8	0.0380359

Nápad 2 - naprogramované pomocou definovania funkcie

In[ ]:=
```
Clear[m, n, P, σ, g, lmin, lmax, krok, Mk, Mo, Mor, d]
m = 300;
n = 50;
P = 5.5 * 10^3;
σ = 1.07 * 10^8;
g = 9.81;

  Mk = P / (2 Pi * n);
  Mo = m * g * l / 4;
  Mor = Sqrt[Mo^2 + 0.75 Mk^2];
  d[l_] = (10 Mor / σ) ^ (1 / 3);


lmin = 0.4;
lmax = 0.8;
krok = 0.05;

vysledok = Table[{l , d[l]}, {l, lmin, lmax, krok}];
TableForm[vysledok // N, TableHeadings → {None, {"l[m]", "d [m]"}},
          TableSpacing → {1, 4}]
```

Out[ ]//TableForm=

l[m]	d [m]
0.4	0.0301991
0.45	0.0314054
0.5	0.0325258
0.55	0.0335741
0.6	0.0345609
0.65	0.0354944
0.7	0.0363812
0.75	0.0372269
0.8	0.0380359

Príklad na samostatné počítanie

Prvky množiny *H* sú náhodné celé èísla z intervalu <-16.2 , 25.78> a je ich 30. Vygenerujte vašu množinu *H*.

Zostavte program, ktorý vypíše

a) maximálny prvok množiny *H* a jeho poradie v množine *H*.

b) minimálny prvok množiny *H* a jeho poradie v množine *H*.

c) súčet prvkov a počet prvkov (aj v percentách), ktorých hodnota je z intervalu $[\overline{h} - 5, \overline{h} + 5>$, kde $\overline{h}$ je aritmetický priemer množiny H.

Príklad na samostatné počítanie

Postupne sú zadávané hmotnosti ulovených ryb v gramech. Zadaním nulovej váhy se ukončí zadávanie. Vytlačte celkovú hmotnosť úlovku. Program má komunikovať s užívateľom.

Príklad na samostatné počítanie

Naprogramujte vyhľadanie najväčšieho spoločného deliteľa dvoch prirodzených čísel pomocou Eulidovho algoritmu. Ak algoritmus nepoznáte, pohľadajte si ho na webe.

Príklad na samostatné počítanie

Vypočítajte n-tú odmocninu z nezáporného čísla a pomocou Newtonovho algoritmu. Použite iteračný predpis $x_{n+1} = x_n - \frac{{x_n}^k}{k\,{x_n}^{k-1}}$

Úlohy na samostatné počítanie

Vytvorte hru "Hádaj, na aké číslo myslím".
Program vygeneruje náhodné číslo a vy ho budete hádať zadávaním hodnôt pomocou príkazu Input. Program bude vypisovať, či máte ubrať alebo pridať, zároveň vám bude počítať pokusy a na záver vám napíše, na ktorý pokus ste vymyslené číslo uhádli.

Príklad na samostatné počítanie

Napíšte program, ktorý simuluje nasledujúci výpočet. Predpokladajme, že číslo π vydelíme dvomi. Výsledné číslo opäť vydelíme dvomi. Tento proces opakujeme až pokiaľ nedostaneme číslo, ktoré je menšie alebo rovné 0.01. Koľko iterácií potrebujeme?

Príklad na samostatné počítanie

Vypočítajte koreň rovnice $e^{-x^2} = 2\,x^2 - 3\,x$ pomocou Newtonovej metódy.
Newtonova metóda vyžaduje voľbu štartovacieho bodu x_0 a následné použite iteračného predpisu $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
Pomôcka: Ak funkciu f definujete pomocou predpisu f [x_] = ..., potom jej deriváciu vypočítate pomocou f '[x].

```
In[ ]:= f[x_] = Sin[x]
Out[ ]= Sin[x]

In[ ]:= f'[x]
Out[ ]= Cos[x]
```

Príklad na samostatné počítanie

Vytvorte množinu A tvorenú 12 náhodných reálnych čísel z intervalu 1, 10. Zistite

1. počet čísel z danej množiny, ktorých hodnota je menšia ako 5,

2. súčet čísel z danej množiny, ktorých hodnota je menšia ako 5 .
3. počet a súčet čísel z danej množiny, ktorých hodnota je menšia ako 5 .
1. počet čísel z danej množiny, ktorých hodnota je menšia ako 5

Nápad - pomocou Pure Function - pre programátorov

```
In[ ]:= Select[A, # < 5 &]
        Select[A, # < 5 &] // Length
Out[ ]= {1.61683, 2.8867, 3.10697, 1.03664, 4.17648, 4.16404}
Out[ ]= 6
```

Nápad - pomocou Pattern Matching (definujete vzor) - pre programátorov

```
In[ ]:= testQ[x_] = If[x < 5, True, False];
        Select[A, testQ]
        Select[A, testQ] // Length
Out[ ]= {1.61683, 2.8867, 3.10697, 1.03664, 4.17648, 4.16404}
Out[ ]= 6
```

Príklad na samostatné počítanie

Vypočítajte druhú odmocninu z nezáporného čísla a pomocou Newtonovho algoritmu. Použite iteračný predpis $x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$

Konkrétne napríklad $\{x_{n+1}\}$ vypočítaná podľa vzťahov $x_{n+1} = 1/2\left(x_n + \frac{2}{x_n}\right)$ bude konvergovať (blížiť sa) ku číslu $\sqrt{2}$.

Konkrétne napríklad $\{x_{n+1}\}$ vypočítaná podľa vzťahov $x_{n+1} = 1/2\left(x_n + \frac{5}{x_n}\right)$ bude konvergovať (blížiť sa) ku číslu $\sqrt{5}$.

Nápad - použijeme rekurzívne programovanie

Rekurzívne programovanie je založené na viac-násobnom opakovaní tej istej funkcie na výsledok získaný v predchádzajúcom výpočte

To znamená, že vlastne realizujeme výpočet $\frac{1}{2}\left(\frac{1}{2}\left(x_1 + \frac{2}{x_1}\right) + \frac{2}{\frac{1}{2}\left(x_1 + \frac{2}{x_1}\right)}\right)$....

Definujeme funkciu, ktorá popisuje iteračnú schému

```
In[ ]:= newton[x_] = 1/2 (x + a/x) // N
Out[ ]= 0.5 (2./x + x)
```

Funkcia Nest zabezpečí toľko-násobné vnorenie funkcie do seba, koľko jej zadáme. Musíme ale zadať aj

štartovací bod - v tomto príklade 2

In[ ]:=
```
Nest[newton, 2, 10]
```

Out[ ]=
```
1.41421
```

Funkcia NestList zabezpečí toľko-násobné vnorenie funkcie do seba, koľko jej zadáme. Musíme ale zadať aj štartovací bod - v tomto príklade 2. Na rozdiel od funkcie Nest vypíše aj medzihodnoty.

In[ ]:=
```
NestList[newton, 2, 10]
```

Out[ ]=
```
{2, 1.5, 1.41667, 1.41422, 1.41421, 1.41421, 1.41421, 1.41421, 1.41421, 1.41421, 1.41421}
```

Funkcia FixedPointList zabezpečí toľko-násobné vnorenie funkcie do seba, koľko jej zadáme. Musíme ale zadať aj štartovací bod - v tomto príklade 2. Na rozdiel od funkcie Nest vypíše aj medzihodnoty a automaticky sa zastaví aj rozdiel medzi dvoma po sebe nasledujúcimi hodnotami je "veľmi malý".

In[ ]:=
```
FixedPointList[newton, 2, 10]
```

Out[ ]=
```
{2, 1.5, 1.41667, 1.41422, 1.41421, 1.41421, 1.41421}
```