3. Programiranje pomoću programskog paketa *Mathematica*

3.1. Relacijski i logički operatori

Relacijski operator (operator uspoređivanja) uspoređuje dva broja i utvrđuje da li je iskaz uspoređivanja (npr. $5 < 8$) "istinit" ("točan", engl. *true*) ili "neistinit" ("netočan", engl. *false*). Logički operatori ispituju izraze čije vrijednosti mogu biti true (istinito, točno) ili false (neistinito, netočno). Na primjer logički operator AND (logičko i) daje rezultat true samo ako oba izraza koje ispituje imaju vrijednost true. Relacijski i logički operatori mogu se koristiti u matematičkim izrazima, a često se upotrebljavaju u kombinacijama s drugim operatorima pri donošenju odluka pomoću kojih se upravlja tijekom izvršavanja programa.

Mathematica podržava sljedeće relacijske operatore:

relacijski operator	*opis*
$<$	manje od
$>$	veće od
$<=$	jednako ili manje od
$>=$	jednako ili veće od
$==$	jednako
$!=$	različito od

U matematičkim izrazima koji sadrže relacijske i aritmetičke operatore, aritmetičke operacije ($+, -, *, /$) imaju prioritet nad relacijskim operacijama. Relacijski operatori imaju međusobno jednak prioritet. Redoslijed prioriteta može se promjeniti pomoću zagrada.

Nekoliko primjera:

```
5 > 8

False

5 < 10

True

5 * 3 == 60 / 4

True

3 + 4 < 16 / 2

True
```

Slika 3.1.

Mathematica prepoznaje sljedeće logičke operatore:

logički operator	*naziv*	*opis*
&&	AND	Djeluje na dva operanda (A i B). Ako oba imaju vrijednost true, rezultat je true; u suprotnom rezultat je false.
||	OR	Djeluje na dva operanda (A i B). Ako jedan ili oba imaju vrijednost true, rezultat je true; u suprotnom rezultat je false.
!	NOT	Djeluje na jedan operand (A), i daje vrijednost suprotnu vrijednosti operanda; rezultat je true ako je operand false, a ako je operand true, rezultat je false.

```
x == x && y == y

True

x == x || y ≠ y

True

(1 < 2) && ! (4 < 2)

True
```

Slika 3.2.

Mathematica ima ugrađene logičke funkcije koje su ekvivalentne logičkim operatorima. To su sljedeće funkcije:

And[A,B]	ekvivalentna izrazu A&&B
Or[A,B]	ekvivalentna izrazu A‖B
Not[A]	ekvivalentna izrazu !A
Xor[A,B]	Isključivo ili. Vraća true ako je jedan operand true, a drugi false.

3.2. Uvjetni iskazi

Uvjetni iskaz je naredba koja *Mathematici* omogućava da odluči da li će izvršiti grupu naredbi koja slijedi iskazu za uvjetno izvršavanje, ili će te naredbe preskočiti.

IF-END struktura:

If[uvjetni izraz, grupa naredbi]

Ako uvjetni izraz ima vrijednos true, program izvršva naredbe koje slijede nakon zareza. Ako je uvjetni izraz false, program preskače grupu naredbi iza zareza, i nastavlja s izvršavanjem naredbi nakon].

IF-ELSE-END struktura:

If[uvjetni izraz, grupa naredbi (1), grupa naredbi (2)]

Ako je vrijednost uvjetnog izraza true, program izvršava prvu grupu naredbi nakon zareza, a ako je vrijednost uvjetnog izraza false, program izvršva drugu grupu naredbi (nakon drugog zareza).

Primjer:

```
A = 7;
B = 8;
If[A < B, Print[B], Print[A]]

8

A = 23;
B = 15;
If[A < B, Print[B], Print[A]]

23
```

Slika 3.3.

ELSEIF struktura:

```
If[uvjet(1),  naredbe (1),
              If[uvjet(2),   naredbe (2),   ...
                             If[uvjet(m),   naredbe (m),
                                            naredbe (m+1)
                             ]
              ]
]
```

Ukoliko uvjetni izraz ima vrijednost true, program izvršava prvu grupu naredbi. Ako uvjetni izraz u iskazu ima vrijednost false, program prelazi na novi iskaz If. Ako uvjetni izraz u ovom iskazu ima vrijednost true, program izvršava drugu grupu naredbi, itd. Na kraju, ako uvjetni izraz u m-tom If iskazu ima vrijednost true, program izvršava m-tu grupu naredbi, a ako ima vrijednost false izvršava (m+1) grupu naredbi.

Primjer 3.1. *Napišite program za algoritam iz Primjera 2.3..*

Rješenje.

```
a = 2; b = 5; c = 2;
DS = b^2 - 4 * a * c;
If[DS > 0,
  {x1 = (-b - Sqrt[DS])/(2 * a), x2 = (-b + Sqrt[DS])/(2 * a), Print["x1=", x1, " ; ", "x2=", x2]},
  If[DS == 0,
    {x = -b/(2 * a), Print["jedinstveno rješenje: ", x]},
    Print["nema realnih rješenja"]
  ];
 ];
```

```
x1=-2 ; x2=-1/2
```

Slika 3.4.

```
a = 1;
b = 2;
c = 1;
DS = b^2 - 4 * a * c;
If[DS > 0,
  {x1 = (-b - Sqrt[DS])/(2 * a), x2 = (-b + Sqrt[DS])/(2 * a), Print["x1=", x1, " ; ", "x2=", x2]},
  If[DS == 0,
    {x = -b/(2 * a), Print["jedinstveno rješenje: ", x]},
    Print["nema realnih rješenja"]
   ];
 ];
```

```
jedinstveno rješenje: -1
```

Slika 3.5.

```
a = 1;
b = 2;
c = 3;
DS = b^2 - 4 * a * c;
If[DS > 0,
  {x1 = (-b - Sqrt[DS])/(2 * a), x2 = (-b + Sqrt[DS])/(2 * a), Print["x1=", x1, " ; ", "x2=", x2]},
  If[DS == 0,
    {x = -b/(2 * a), Print["jedinstveno rješenje: ", x]},
    Print["nema realnih rješenja"]
   ];
 ];
```

```
nema realnih rješenja
```

Slika 3.6.

3.3. Petlje

Mathematica podržava dvije strukture petlji: DO i DOWHILE.

DO struktura:

Do[tijelo petlje,{k, kmin, kmax, korak}]

Tijelo petlje izvršava se sve dok k poprima cjelobrojne vrijednosti od *kmin* do *kmax* s zadanim *korakom*.

Primjer uporabe petlje DO:

```
Do[{x = k^2, Print[x]}, {k, 1, 10, 3}]
1
16
49
100
```

Slika 3.7.

DOWHILE struktura:

For[početak, uvjetni izraz, korak, tijelo petlje]

Tijelo petlje izvršava se sve dok je vrijednost uvjetnog izraza true.

Može se koristiti i naredba:

While[uvjetni izraz, tijelo petlje]

Uvjetni izraz u naredbi While mora sadržavati barem jednu varijablu, čija vrijednost mora biti poznata kad *Mathematica* prvi put izvši naredbu While. Unutar tijela petlje mora postojati barem jedna naredba koja dodjeljuje novu vrijednost barem jednoj od tih varijabli. U suprotnom, izvršavanje petlje se nikad ne bi zaustavilo, zato što bi uvjetni izraz uvijek imao vrijednost true.

Primjer jednostavnih petlji For i While prikazan je u narednom programu, u kojem vrijednost varijable x počinje od 1 i udvostručuje se u svakom ponavljanju petlje sve dok je x manje ili jednako 15.

```
For[x = 1, x ≤ 15, x = 2 * x, Print[x]]
1
2
4
8

x = 1;
While[x ≤ 15, {Print[x], x = x * 2}]
1
2
4
8
```

Slika 3.8.

Primjer 3.2. *Napišite program za algoritam iz Primjera 2.4..*

Rješenje.

```
N1 = 13;
Do[{L = K^2, Print[K, " ; ", L]}, {K, 1, N1, 2}]
1 ; 1
3 ; 9
5 ; 25
7 ; 49
9 ; 81
11 ; 121
13 ; 169
```

Slika 3.9.

```
In[1]:= N1 = 13;
        For[K = 1, K ≤ N1, K = K + 2, {L = K^2, Print[K, " ; ", L]}]

        1 ; 1

        3 ; 9

        5 ; 25

        7 ; 49

        9 ; 81

        11 ; 121

        13 ; 169
```

Slika 3.10.

```
N1 = 13;
K = 1;
While[K ≤ N1, {L = K^2, Print[K, " ; ", L], K = K + 2}]

1 ; 1

3 ; 9

5 ; 25

7 ; 49

9 ; 81

11 ; 121

13 ; 169
```

Slika 3.11.