

Listy

Lista n -elementowa, to obiekt postaci $\{a_1, a_2, \dots, a_n\}$

`Part[[lista,i]]` lub `lista[[k]]` to k -ty element listy (liczymy od 1), jeśli $k < 0$, to $|k|$ -ty element listy licząc od końca

`Length[lista]` zwraca długość listy

Sposoby definiowania listy:

1. `Table[f[i], {i,n}]` tworzy listę $\{f[1], \dots, f[n]\}$;
2. `Table[f[i], {i,0,n}]` tworzy listę $\{f[0], \dots, f[n]\}$;
3. `Table[f[i], {i,n,m}]` tworzy listę $\{f[n], f[n+1], \dots, f[m-1], f[m]\}$;
4. `Table[f[i], {i,imin,imax,istep}]` tworzy listę $\{f[imin], f[imin+istep], f[imin+2*step], \dots, f[imax]\}$;

Listy liczb można też tworzyć za pomocą funkcji `Range`:

1. `Range[n]` tworzy listę $\{1, 2, \dots, n\}$;
2. `Range[n1,n2]` tworzy listę $\{n1, n1+1, \dots, n2-1, n2\}$; and
3. `Range[n1,n2,ntstep]` tworzy listę $\{n1, n1 + nstep, n1 + 2*nstep, \dots, n2-nstep, n2\}$

Wiele funkcji w Mathematicie posiada atrybut `Listable`, co oznacza, że ich działanie jest automatycznie rozszerzane do całych list. Przykładowo, możemy napisać: `Sqrt[1+Range[10]^2]` Ten sam efekt uzyskamy pisząc: `Table[Sqrt[1+n^2],{n,10}]`

Uwaga: Pisząc funkcje działające na listach stosuj `:=` zamiast `=`

Zadanie 1. Wygeneruj listę stu jedynek, tzn. $\{1, 1, \dots, 1\}$

Zadanie 2. Wygeneruj na dwa sposoby listę $\{1, 2, \dots, 100\}$

Zadanie 3. Wygeneruj na dwa sposoby listę $\{2, 4, \dots, 100\}$

Zadanie 4. Wygeneruj na dwa sposoby listę $\{1, 3, \dots, 99\}$

Zadanie 5. Wygeneruj na dwa sposoby listę $\{1, 1/2, 1/3, \dots, 1/100\}$

Zadanie 6. Wygeneruj na dwa sposoby listę pierwszych stu liczb Fibonacciego.

Wskazówka: Wartością funkcji `Fibonacci[n]` jest n -ta liczba Fibonacciego.

Zadanie 7. Wygeneruj na dwa sposoby listę 100 pierwszych liczb pierwszych.

Wskazówka: Wartością funkcji `Prime[n]` jest n -ta liczba pierwsza.

Zadanie 8. Napisz funkcję, której argumentem jest lista, a wartością suma elementów tej listy, tzn. twoja funkcja ma robić to samo co funkcja wbudowana `Total`

Zadanie 9. Napisz funkcję, której argumentem jest lista, a wartością iloczyn elementów tej listy.

Zadanie 10. Napisz funkcję, której argumentem jest lista, a wartością jest lista bez ostatniego elementu, tzn. twoja funkcja ma robić to samo co wbudowana funkcja **Most**

Zadanie 11. Napisz funkcję, której argumentem jest lista, a wartością jest lista bez pierwszego elementu, tzn. twoja funkcja ma robić to samo co wbudowana funkcja **Rest**

Zadanie 12. Napisz funkcję, której argumentem jest lista, a wartością lista z odwróconą kolejnością elementów, tzn. twoja funkcja ma robić to samo co funkcja wbudowana **Reverse**

Zadanie 13. Wiedząc, że **Join** łączy dwie (lub więcej) listy w jedną, wygeneruj na co najmniej trzy różne sposoby listę $\{1, 2, \dots, 20, 20, 99, \dots, 1\}$

Zadanie 14. Wygeneruj na co najmniej trzy różne sposoby listę $\{1, 2, \dots, 19, 20, 19, \dots, 1\}$

Zadanie 15. Napisz funkcję, która dodaje dwie listy (element po elemencie), czyli robi to samo co operator plus. Jak zachowuje się twoja funkcja dla list różnej długości? Jak zachowuje się dla takich list operator plus?

Zadanie 16. Wygeneruj listę długości 100 postaci $\{1, -1, 1, -1, \dots, 1, -1\}$

Zadanie 17. Wygeneruj listę długości 100 postaci $\{1, -2, 3, -4, \dots, 99, -100\}$ bezpośrednio, oraz korzystając z mnożenia list, tzn. wymnóż (operator kropki) listę $\{1, 2, \dots, 100\}$ przez listę $\{1, -1, 1, -1, \dots, 1, -1\}$.

Wykresy ciągów

Wykres ciągu punktów z listy jest tworzony przy pomocy funkcji **ListPlot**.

1. **ListPlot** $\{\{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_n, y_n\}\}$ tworzy wykres listy punktów

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n).$$

2. **ListPlot** $\{y_1, y_2, \dots, y_n\}$ tworzy wykres listy punktów

$$(1, y_1), (2, y_2), \dots, (n, y_n).$$

Zadanie 18. Wyświetl wykres ciągu $(\sin n)$, $n = 1, 2, \dots, 1000$.

Zadanie 19. Niech p_n będzie n -tą liczbą pierwszą. Wyświetl punkty ciągu $(p_{n+1} - p_n)$, $n = 1, 2, \dots, 1000$.

Lista list

`t1 = Table[a[i, j], {i, 2, 10, 2}, {j, 1, 5}]` daje w wyniku listę list. Aby wyświetlić tę listę w czytelny sposób napisz `TableForm[t1]` Ważne, aby `TableForm` używać tylko przy wyświetlaniu tablic, a nie przy ich definiowaniu!

Każdy indeks może zależeć od poprzedniego. Przykładowo możesz napisać

```
Table[1, {i, 10}, {j, i}] // TableForm
```

`Part[lista, {i, j}]` lub `lista[[i, j]]` zwraca element j -ty w i -tej podliście.

Zadanie 20. Wygeneruj tabliczkę mnożenia do 100 (tzn. od $1 \cdot 1$ do $10 \cdot 10$).

Zadanie 21. Korzystając z faktu, że `Binomial[n,m]` zwraca $\binom{n}{m}$ wygeneruj trójkąt Pascala (np. 7 wierszy). Ma to być trójkąt, czyli w tablicy ma nie być żadnych zer.

Zadanie 22. Napisz funkcję, która przeplata dwie listy, przykładowo: `przeplot[{1,2,3},{10,11,12}]` ma zwrócić listę `{1,10,2,11,3,12}`. Możesz założyć, że listy są tej samej długości.

Wsk. Skorzystaj z funkcji `Flatten`, która „spłaszcza” listę, przykładowo dla listy

`x={ {1,2,3}, {4,{5},{6,7}}, {{{42}}} }`

`Flatten[x]` zwraca listę `{1,2,3,4,5,6,7,42}`

Zadanie 23. Jeszcze raz wygeneruj listę długości 100 postaci `{1,-2,3,-4,...,99,-100}` Tym razem skorzystaj z funkcji `przeplot`

Zadanie 24. W tym zadaniu (i być może w następnych) skorzystamy z funkcji `If`. Jej działanie wyjaśni poniższy przykład definicji wartości bezwzględnej

`abs[x_] := If[x < 0, -x, x]`

Wygeneruj tablicę:

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

Zadanie 25. Rozwiąż powyższe zadanie korzystając z funkcji `Boole`. Sprawdź za pomocą `?Boole` co ona robi.

Zadanie 26. Wygeneruj tablicę:

```
1 1 1 1 1
0 1 1 1 1
0 0 1 1 1
0 0 0 1 1
0 0 0 0 1
```

Zadanie 27. Wygeneruj tablicę:

```
1 2 3 4 5
0 1 2 3 4
0 0 1 2 3
0 0 0 1 2
0 0 0 0 1
```

Zadanie 28. Wygeneruj tablicę:

```
1 1 1 1 1
1 1 1 1 0
1 1 1 0 0
1 1 0 0 0
1 0 0 0 0
```

Zadanie 29. Wygeneruj tablicę:

```

1 2 3 4 5
2 3 4 5 0
3 4 5 0 0
4 5 0 0 0
5 0 0 0 0

```

Zadanie 30. Wygeneruj tablicę:

```

1   2   3   4   5
6   7   8   9  10
11  12  13  14  15
16  17  18  19  20
21  22  23  24  25

```

Zadanie 31. Wygeneruj tablicę:

```

1   6  11  16  21
2   7  12  17  22
3   8  13  18  23
4   9  14  19  24
5  10  15  20  25

```

Zadanie 32. Napisz funkcję, która będzie transponowała tablicę. Oczywiście istnieje taka funkcja wbudowana `Transpose`, ale w tym zadaniu masz z niej nie korzystać.

Zadanie 33. Napisz funkcję, która odwróci kolejność wierszy tablicy.

Zadanie 34. Napisz funkcję, która odwróci kolejność kolumn tablicy.

Zadanie 35. Napisz funkcję, która będzie obracała prostokątną tablicę o 90 stopni w lewo. Przykładowo, dla tablicy:

```

1 2 3
4 5 6

```

funkcja zwróci tablicę

```

3 6
2 5
1 4

```

Zadanie 36. Napisz funkcję, która będzie obracała prostokątną tablicę o 90 stopni w prawo.

Zadanie 37. Napisz funkcję, która będzie obracała prostokątną tablicę o 180 stopni.

Zadanie 38. Napisz funkcję, której wartością będzie i -ty wiersz danej tablicy.

Zadanie 39. Napisz funkcję, której wartością będzie j -ta kolumna danej tablicy prostokątnej.

Zadanie 40. Napisz funkcję, która dla zadanego k zwraca tablicę o wymiarach $k \times k$ tak, aby w i -tym wierszu i j -tej kolumnie stało 1 jeśli $\text{GCD}[i, j] = 1$ i 0 w przeciwnym razie. Wyświetl tę tablicę dla $k = 50$ za pomocą polecenia `ArrayPlot[t]`

Zadanie 41. Wygeneruj i wyświetl szachownicę.

Zadanie 42. To samo, co w poprzednim zadaniu, ale niech białe pola staną się czarnymi, a czarne - białymi.

Zadanie ()** Jeśli dotarłeś do tego miejsca i nie masz co robić, to spróbuj wygenerować i wyświetlić spiralę Ulama (patrz https://pl.wikipedia.org/wiki/Spirala_Ulama) Czarny kwadrat oznacza liczbę pierwszą. *Uwaga:* Najtrudniejsze jest utworzenie spiralnej tablicy liczb. Potem wystarczy zastosować funkcję `PrimeQ`