

Listy - część druga

IntegerDigits

W zadaniach poniżej wykorzystasz funkcję `IntegerDigits`, która zwraca listę cyfr danej liczby.

Zadanie 1. Znajdź liczbę cyfr liczby 2^{128} .

Zadanie 2. Znajdź sumę cyfr liczby 2^{128} .

Zadanie 3. Znajdź iloczyn cyfr liczby 2^{128} .

Zadanie 4. Znajdź liczbę cyfr 7 liczby 2^{128} . Wykorzystaj funkcję `Count`

Zadanie 5. Znajdź liczbę cyfr parzystych liczby 2^{128} . Wykorzystaj funkcję `Count`

Zadanie 6. Znajdź pierwszych dziesięć cyfr liczby 2^{128} . Wykorzystaj funkcję `Take`

Zadanie 7. Znajdź ostatnich dziesięć cyfr liczby 2^{128} . Wykorzystaj funkcję `Take`

Zadanie 8. Znajdź największą cyfrę liczby 2^{20}

Zadanie 9. Utwórz listę list cyfr pierwszych dziesięciu kwadratów.

Zadanie 10. Znajdź liczbę zer na końcu liczby $100!$. Wykorzystaj funkcję `Split`

ListPlot i ListLinePlot

`ListLinePlot` łączy kolejne punkty, `ListPlot` tego nie robi.

Zadanie 11. Utwórz wykres (`ListLinePlot`) liczby cyfr pierwszych stu kwadratów.

Zadanie 12. Utwórz wykres (`ListLinePlot`) wartości pierwszej cyfry pierwszych stu kwadratów.

Zadanie 13. Narysuj wykres (`ListPlot` lub `ListLinePlot`) liczby zer na końcu liczby $n!$ dla n od 1 do 200.

Zadanie 14. Utworzyć wykres (`ListPlot`) zbioru punktów

$$(\sin(i + j), \sin(i - j)), \quad i = 1, 2, \dots, 100, \quad j = 1, 2, \dots, 100,$$

Wskazówka: Wykorzystaj `Flatten[lista, 1]`

Wyjaśnienie: Korzystając z funkcji `Flatten` możemy spłaszczyć listę o wybraną liczbę poziomów. Poniższe przykłady powinny wyjaśnić o co chodzi.

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

Select

`Select[lista, warunek]` tworzy listę tych elementów e listy `lista`, dla których `warunek[e]` ma wartość `True`.

Przykład

```
h[x_] = x > 2
```

```
In[76]:= Select[{1, 2, 4, 7, 6, 2}, h]
```

```
Out[76]= {4, 7, 6}
```

Zadanie 15. Znaleźć najmniejszą liczbę pierwszą w ciągu

$$10^{10} + 1, 10^{10} + 2, 10^{10} + 3, \dots$$

Rozwiązanie.

Najpierw należy za pomocą polecenia

```
t = Table[10^10 + n, {n, 1, 100}];
```

zdefiniować listę 100 pierwszych wyrazów rozpatrywanego ciągu.

Następnie polecenie:

`Select[t, PrimeQ]` wyświetli te liczby n z listy `t`, dla których `PrimeQ[n]` daje wartość prawdy.

Zadanie 16. 11 jest liczbą pierwszą. Znaleźć dwie następne liczby pierwsze, w zapisie których występują same jedynki.

Rozwiązanie:

Zdefiniujemy najpierw rekurencyjnie funkcję

$$f(n) = \sum_{k=0}^n 10^k.$$

```
f[1] := 1; f[n_] := 10*f[n - 1] + 1;
```

Sprawdźmy pisząc `f(5)`. Wynikiem powinna być liczba 11111.

Utwórzmy teraz listę 30 pierwszych liczb „jedynekowych”:

```
t = Table[f[n], {n, 1, 30}];
```

I wreszcie:

```
wynik = Select[t, PrimeQ].
```

Ile jedynek jest w zapisie trzeciej liczby?

```
Length[IntegerDigits[ wynik[[3]] ]]
```

Zadanie 17. Ponownie znajdź liczbę cyfr parzystych liczby 2^{128} . Wykorzystaj predykat `EvenQ`

Zadanie 18. Znajdź wszystkie liczby pierwsze, wśród 100 początkowych liczb Fibonacciego.

Funkcje anonimowe

Jeżeli funkcji używamy tylko raz, można ją zdefiniować „w locie”, jako tzw. funkcję anonimową.

Wyrażenie `# > 2 &` definiuje tę samą funkcję, co `h[x_] = x > 2`. Znak `#` oznacza argument, znak `&` kończy definicję funkcji. Sprawdźmy

```
In[78]:= (# > 2) &[6]
```

```
Out[78]= True
```

Można więc napisać

```
In[76]:= Select[{1, 2, 4, 7, 6, 2}, (# > 2)&]
```

```
Out[76]= {4, 7, 6}
```

Zadanie 19. Wyznaczyć takie liczby $n \in \{2, 3, \dots, 1000\}$, które spełniają warunek:

$$2^{n-1} = 1 \pmod n$$

i są liczbami złożonymi (takie liczby nazywają się pseudopierwsze przy podstawie 2).

Rozwiązanie. Tworzymy listę liczb $2, 3, \dots, 1000$: `t = Table[n, {n, 2, 1000}]`;

Teraz zastosujemy funkcję `Select[t, warunek]`.

Ostatecznie napiszmy:

```
Select[t, PowerMod[2, # - 1, #] == 1 && ! PrimeQ[#] &]
```

co oznacza: wybierz te elementy `#` z listy `t`, dla których

$$2^{\#-1} = 1 \pmod n \quad \text{oraz nie prawda, że } \# \text{ jest pierwsze}$$

Zadanie 20. Ile jest liczb pseudopierwszych przy podstawach 2, 3 i 5 jednocześnie, mniejszych od miliona?

Map i Apply

1. `Map[f, {a1, ..., an1}]`

lub

`f /@ {a1, ..., an1}`

Wynik: lista wartości `{f[a1], ..., f[an]}`

Przykład:

```
In[1]:= Map[f, {a, b, c, d, e}]
```

```
Out[1]= {f[a], f[b], f[c], f[d], f[e]}
```

Przypomnijmy, że wiele funkcji w Mathematicie jest automatycznie „mapowana” na elementy listy (tzn. posiada atrybut `Listable`)

2. `Apply[f, {a1, ..., an1}]`

lub

`f @@ {a1, ..., an1}`

Wynik: zastąpienie nagłówka listy przez `f`

Przykład:

```
In[81]:= Times @@ {a, b}
```

```
Out[81]= a b
```

Wyjaśnienie: pełna forma zapisu $\{a, b\}$ to `List[a, b]`. `List` jest nagłówkiem. W wyniku `Times @@ {a, b}` nagłówek ten zostaje zastąpiony przez `Times`, tzn. „Razy”. Otrzymujemy `Times[a, b]`, czyli iloczyn $a \cdot b$. Podobnie `Superscript @@ {a, b}` daje wynik a^b (nie jest potęgą, ale sposób napis, w którym drugi znak występuje w pozycji indeksu górnego).

3. `Apply[f, {a1, ..., an1}, {1}]`

lub

`f @@@ {a1, ..., an1}`

Wynik: zastąpienie nagłówka każdego elementu przez `f`

Zadanie 21. Rozłożyć na czynniki pierwsze liczbę $20!$.

Wsk. najpierw napisać

`wynik = FactorInteger[ 20! ]`,

a potem

`CenterDot @@ (Superscript @@@ wynik)`.

Objaśnienie: `Superscript @@@ wynik` stosuje funkcję `Superscript` (zapis typu x^y) po kolei do każdego elementu listy `wynik`, co daje listę:

$$\{2^{18}, 3^8, 5^4, 7^2, 11^1, 13^1, 17^1, 19^1\}$$

Teraz zastosowanie `CenterDot` do tej listy powoduje wstawienie kropki między elementami tej ostatniej listy.

Zadanie 22. Wymnóż elementy listy za pomocą `Apply`

Zadanie 23. Znajdź największy element listy za pomocą `Apply`