Technologie informacyjne
przed 7 wykładem

Andrzej Giniewicz

10.04.2024

Dzisiejsza$ porcja zajęć z LaTeXa dotyczy wzorów matematycznych. W notatkach poruszymy zasady składu w języku polskim i angielskim. Jeśli składamy tekst w innym języku, upewnijmy się, jaka jest tradycja typograficzna danego kraju.

1 Rekomendowane pakiety

Do preambuły dodajemy zestaw pakietów do składu tekstów matematycznych.

```
\usepackage{mathtools,amsthm,amssymb,icomma,upgreek,xfrac}
\mathtoolsset{showonlyrefs,mathic}
```

Pakiety te i opcje zostaną opisane w odpowiednich miejscach poniżej.

2 Liczby w tekście

Zacznijmy od zapisu liczb, który mocno zależy od ich użycia. Po pierwsze, są wartości, które mogą być z powodzeniem zastąpione liczebnikami. W takim przypadku traktujemy liczby jako skrótowy zapis liczebników i stosujemy zasady tekstowe i krój pisma taki jak w tekście. W szczególności pamiętajmy o stawianiu kropek po liczebnikach porządkowych. Jeśli font wspiera tak zwane cyfry nautyczne[1], które w większości mieszczą się pomiędzy linią środkową a bazową pisma, czyli mają wysokość równą wysokości-x (0123456789), stosujemy je do zapisu takich właśnie liczebników, ponieważ ich wysokość nie wytrąca nas z płynności czytania. Cyfr nautycznych użyjemy na przykład do zapisu roku 2023. Dla odmiany, jeśli liczba ta byłaby wynikiem działania matematycznego, napiszemy 2023. Nie wszystkie fonty wspierają cyfry nautyczne. W trybie tekstowym liczby składamy, wpisując kolejne cyfry. Jeśli liczba jest większa lub równa 10 000, stosujemy tak zwany separator grup, który jest wąskim, niełamliwym odstępem. Odstęp ten w trybie tekstowym wstawiamy ręcznie co trzy miejsca, licząc od cyfry jedności w lewo. Separatorów grup nie używamy dla liczebników używanych w roli referencji, czyli na przykład numerów stron, numerów norm i temu

[1] Patrz http://www.texfaq.org/FAQ-osf.

podobnych. Odstęp odpowiedzialny za separator grup składamy w LaTeXu komendą `\,`, czyli liczbę 1 234 567 wprowadzimy jako `1\,234\,567`. Jeśli liczba ma część ułamkową, stawiamy separator dziesiętny i wypisujemy kolejne cyfry po przecinku. Do cyfr po przecinku nie stosujemy separatorów grup. W języku polskim separatorem dziesiętnym jest przecinek, w języku angielskim kropka. Liczbę 1 234 567,1234567 zapiszemy jako `1\,234\,567,1234567`. Jeśli zapisujemy liczby w postaci ułamków, w trybie tekstowym warto użyć komendy `\sfrac` z pakietu `xfrac`. W ułamkach tego typu nie używa się cyfr nautycznych. Aby złożyć ułamek ½, napiszemy `\sfrac{1}{2}`. Ułamki tego typu bardzo często używane są w przepisach kulinarnych. Jeśli ułamek niewłaściwy zapisujemy w formie liczby mieszanej, czyli liczby całkowitej i ułamka właściwego, oddzielamy je niewielkim odstępem, równym separatorowi grupy, czyli 3 ½ złożymy za pomocą `3\,\sfrac{1}{2}`. Podkreślmy raz jeszcze, że zasady te stosuje się do wartości w trybie tekstowym, czyli w sytuacji, gdy bez straty sensu, liczbę możemy zastąpić liczebnikiem, pisząc na przykład „pół szklanki mąki” zamiast „½ szklanki mąki”.

3 Podstawowe wyrażenia w trybie matematycznym

Gdy liczby i symbole reprezentują operacje matematyczne, nie stosujemy do nich zasad składu tekstowego, tylko matematyczne. Tryb matematyczny ma dwa style — liniowy i wyróżniony. Tryb liniowy służy do składu wzorów wewnątrz tekstu, natomiast tryb wyróżniony do wzorów znajdujących się w osobnych liniach. Wzory w stylu liniowym muszą być nieco niższe, aby swoją wysokością nie wymusić zwiększenia interlinii ponad i pod linią, w której się znajdują. W niniejszych notatkach przedstawimy jedynie zalecane sposoby wchodzenia do trybu matematycznego, pomimo iż w Internecie można natrafić na wiele innych metod.

Aby przejść do trybu matematycznego liniowego, wyrażenie otaczamy nawiasami z symbolem wstecznego ukośnika, czyli `\(...\)`, na przykład, aby uzyskać 2023, napiszemy `\(2023\)`. Pakiet `mathtools`, ładujący dodatkowo pakiet `amsmath`, udostępnia też środowisko do składu wzorów wyróżnionych. Aby uzyskać wyrażenie

$$2023$$

piszemy

```
\begin{equation}
  2023
\end{equation}
```

Gdy użyjemy opcji `showonlyrefs` w komendzie `\mathtoolsset`, równanie nie będzie numerowane, tak jak w przykładzie powyżej. Opcja ta pozwala na automatyczne numerowanie tylko tych równań, do których się odnosimy. Aby dodać numerację do równania, korzystamy z komendy `\label`

```
\begin{equation}\label{year-equation}
  2023
```

```
\end{equation}
```

po czym do odwołania się do równania, piszemy na przykład

```
równanie \eqref{year-equation} na stronie \pageref{year-equation}
```

Jeśli utworzymy etykietę jak w przykładzie i odwołamy się do równania (1), obok niego pojawi się numer

$$2023. \tag{1}$$

Jeśli z jakiegoś powodu nie możemy używać opcji `showonlyrefs`, wszystkie wzory złożone za pomocą środowiska `equation` będą numerowane. Jest to zła praktyka, której należy unikać — tylko wzory, do których się odnosimy, powinny mieć numery. W takiej sytuacji we wzorach, w których nie chcemy, aby były numerowane, zamieniamy środowisko na `equation*`, co spowoduje bezwarunkowe wyłączenie numeracji niezależnie od sytuacji.

Zwróćmy uwagę na interpunkcję. Interpunkcję po wzorach stawiamy tak, jak przy słowach użytych do przeczytania wzoru. Oznacza to, że posługujemy się przecinkami i kropkami. Nie stawiamy dwukropka przed pojedynczym wzorem, tak samo jak nie piszemy „Niebo było: niebieskie.” — użyjemy go natomiast, jeśli wyliczamy kilka wzorów jeden po drugim, pamiętając jednocześnie o stawianiu przecinków po każdym wypisanym wzorze oprócz ostatniego, który otrzyma kropkę.

W trybie matematycznym do zapisu liczb nie stosujemy separatorów grup. Niestety domyślnie LaTeX stosuje konwencję angielską, w której kilka liczb w ciągu oddziela się przecinkami, więc zastosowanie przecinka powoduje zwiększenie odstępu — komenda `\((3,14, 2,78)\)` daje w wyniku $(3, 14, 2, 78)$, co jest równoważne ciągowi czterech liczb. Aby rozwiązać ten problem, można zastosować pakiet `icomma`, który zmienia zachowanie przecinka w LaTeXu na takie, że pomijany jest po nim odstęp, jeśli przecinek jest umieszczony pomiędzy cyframi. Po załadowaniu pakietu `icomma`, ten sam kod da w wyniku $(3,14, 2,78)$. Porównajmy te liczby na zbliżeniu:

$$(3, 14, 2, 78),$$

$$(3,14, 2,78).$$

W pierwszym zapisie odstęp po każdym przecinku jest identyczny. W drugim zapisie odstęp oddzielający liczby jest większy. Zastosowanie pakietu `icomma` nie zawsze jest wystarczające. Niekiedy dodatkowo konieczna jest zamiana przecinków na średniki. Zarazem norma, jak i tradycja typograficzna dopuszcza taką podmianę, jeśli z powodu dwuznaczności przecinka tracimy jednoznaczność lub czytelność. W powyższym przykładzie moglibyśmy napisać `\((3,14; 2,78)\)` uzyskując $(3{,}14; 2{,}78)$, co po powiększeniu wygląda o wiele lepiej

$$(3{,}14; 2{,}78).$$

Zamiany przecinka na średnik używamy wtedy, gdy nie ma jednoznaczności. Jeśli w parze znajdują się zmienne lub inne symbole, nie ma potrzeby podmieniania separatora.

Ważne, aby całe wyrażenie było umieszczone w trybie matematycznym. Oznacza to, że nie piszemy na przykład `\(2\) \(+\) \(3\)`, tylko od razu `\(2 + 3\)`. Spacje wokół operatorów są ignorowane, po czym wstawiane są prawidłowe odstępy wynikające z faktu, że umieszczamy w danym miejscu konkretne symbole.

W trybie matematycznym ułamki składamy za pomocą komendy `\frac`, która zapewnia, że licznik i mianownik są oddzielone poziomą kreską, napiszemy na przykład `\(\frac{1}{12}\)`, co da w wyniku $\frac{1}{12}$. Ułamki możemy zagnieżdżać, na przykład tak jak w kodzie poniżej

```
\begin{equation}
  \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{\ldots}}}}.
\end{equation}
```

Efektem działania tej komendy jest

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{\ldots}}}}.$$

Tego typu ułamki, czyli ułamki łańcuchowe, możemy też złożyć za pomocą komendy `\cfrac`, jeśli kolejne liczby stają się nieczytelne przez swój rozmiar

```
\begin{equation}
 \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{\ldots}}}}.
\end{equation}
```

Efektem działania tej komendy jest

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{\ldots}}}}.$$

Jeśli przechodzimy ze składu liczb do składu z wykorzystaniem symboli, norma nakazuje skład zmiennych oraz stałych i nazw zależnych od kontekstu pismem pochyłym, na przykład f, x, a. Jeśli składamy stałe uniwersalne, na przykład liczbę Eulera e, liczbę pi π, jednostkę urojoną i, funkcję sinus sin, używamy pisma prostego zamiast pochyłego (nie piszemy e, π, i, sin). Pisma prostego używamy również do operatorów, na przykład do literki „d” w zapisie pochodnej. Domyślnie, jeśli używamy LaTeXa i wpiszemy w trybie matematycznym `x` lub `sin`, będzie to złożone tekstem pochyłym, jak zmienne. W przypadku x jest to prawidłowy zapis, ale dla sinusa niestety nie. LaTeX posiada szereg komend, do zapisu standardowych funkcji, takich jak sinus. W tym przypadku powinniśmy napisać `\sin` zamiast `sin`.

Do składu liter greckich służą komendy, których nazwy odpowiadają transliteracji greckich liter do języka angielskiego. Komendy trybu matematycznego pozwalające wpisać pochyły symbol. Symbole proste znajdują się w pakiecie upgreek. Zachęcam do poćwiczenia pisowni alfabetu greckiego[2]. Niektóre litery uzyskujemy komendą \mathrm. Niektóre litery mają dwa warianty, jeśli częściej spotykany jest wariant, w tabeli poniżej znajduje się wersja z var przed nazwą.

Litera	Komendy	Wygląd
alfa	`\alpha A \upalpha \mathrm{A}`	αAαA
beta	`\beta B \upbeta \mathrm{B}`	βBβB
gamma	`\gamma \Gamma \upgamma \upGamma`	$\gamma \Gamma$γΓ
delta	`\delta \Delta \updelta \upDelta`	$\delta \Delta$δΔ
epsilon	`\varepsilon E \upvarepsilon \mathrm{E}`	εEεE
zeta	`\zeta Z \upzeta \mathrm{Z}`	ζZζZ
eta	`\eta H \upeta \mathrm{H}`	ηHηH
teta	`\vartheta \Theta \upvartheta \upTheta`	$\vartheta \Theta$ϑΘ
jota	`\iota I \upiota \mathrm{I}`	ιIιI
kappa	`\varkappa K \upvarkappa \mathrm{K}`	$\varkappa K$ϰK
lambda	`\lambda \Lambda \uplambda \upLambda`	$\lambda \Lambda$λΛ
mi	`\mu M \upmu \mathrm{M}`	μMμM
ni	`\nu N \upnu \mathrm{N}`	νNνN
ksi	`\xi \Xi \upxi \upXi`	$\xi \Xi$ξΞ
omikron	`o O \mathrm{o} \mathrm{O}`	$o O$oO
pi	`\pi \Pi \uppi \upPi`	$\pi \Pi$πΠ
rho	`\rho P \uprho \mathrm{P}`	ρPρP
sigma	`\sigma \Sigma \upsigma \upSigma`	$\sigma \Sigma$σΣ
tau	`\tau T \uptau \mathrm{T}`	τTτT
ipsilon	`\upsilon \Upsilon \upupsilon \upUpsilon`	$\upsilon \Upsilon$υϒ
fi	`\varphi \Phi \upvarphi \upPhi`	$\varphi \Phi$φΦ
chi	`\chi X \upchi \mathrm{X}`	χXχX
psi	`\psi \Psi \uppsi \upPsi`	$\psi \Psi$ψΨ
omega	`\omega \Omega \upomega \upOmega`	$\omega \Omega$ωΩ

Oprócz greki, niekiedy używane są pierwsze cztery litery alfabetu hebrajskiego

Litera	Komenda	Wygląd
alef	`\aleph`	ℵ
bet	`\beth`	ℶ
gimel	`\gimel`	ℷ
dalet	`\daleth`	ℸ

[2] Materiały do druku: `http://prac.im.pwr.edu.pl/~giniew/lib/exe/fetch.php?media=rok2324:letni:greka.pdf`.

Wiele symboli zapisujemy tradycyjnie za pomocą konkretnych krojów pisma. I tak do zbiorów liczbowych użyjemy kroju tablicowego (ang. black board), za pomocą komendy `\mathbb`, na przykład `\mathbb{N}` to $\mathbb{N}$, czyli liczby naturalne. Rodziny zbiorów zapiszemy za pomocą symboli w kroju kaligraficznym komendą `\mathcal`, na przykład `\mathcal{A}` da w wyniku $\mathcal{A}$. Część rzeczywistą liczby zapiszemy frakturą za pomocą komendy `\mathfrak`, na przykład `\mathfrak{Re}(x)` spowoduje wyświetlenie $\mathfrak{Re}(x)$.

Wiele symboli specjalnych również składamy za pomocą komend. Na przykład nieskończoność, `\infty`, ∞ lub zbiór pusty, `\emptyset`, $\emptyset$. W LaTeXu istnieje bardzo dużo symboli i nie sposób wszystkich zapamiętać. Pomocna jest lista symboli wraz z komendami i pakietami, w których są zdefiniowane[3] oraz niewielka aplikacja Internetowa DeTeXify, która pozwala narysować symbol i próbuje odgadnąć, jaką komendą go uzyskać[4].

LaTeX pozwala sterować indeksami górnymi i dolnymi. Jeśli napiszemy `x_i^2`, uzyskamy x_i^2. Jeśli chcemy podnieść wartość do większej potęgi, musimy uważać — `x^12` to $x^1 2$ a nie x^{12}. Aby uzyskać x^{12}, musimy napisać `x^{12}`, czyli użyć grupowania. To samo tyczy się indeksów dolnych. Jeśli w indeksie dolnym znajdują się stałe, na przykład x_n należy się zastanowić, co oznaczają te symbole. Jeśli są zmiennymi, indeksami lub stałymi matematycznymi, indeks napiszemy krojem pochyłym. Jeśli literka „n" jest skrótem dla „new" czyli „nowy", powinniśmy użyć kroju prostego, czyli napisać `x_{\mathrm{n}}`, aby uzyskać x_{n} — dla porównania, jeśli n byłoby indeksem, napiszemy x_n zamiast x_{n}.

LaTeX rozróżnia relacje binarne, operatory binarne i duże operatory. Przykładem relacji binarnej jest =, przykładem operatora binarnego +, natomiast przykładem dużego operatora $\sum$. Standardowo przykładami relacji binarnych są: $<$, $\leqslant$, $=$, $\neq$, $\geqslant$, $>$ uzyskiwane przez `<`, `\leq`, `=`, `\neq`, `\geq`, `>` odpowiednio[5]. Przykładami operatorów są: $+$, $-$, $\cdot$, $\times$ uzyskiwane przez `+`, `-`, `\cdot`, `\times` odpowiednio. Dowolny ciąg znaków możemy przekształcić tak, aby był relacją (`\mathrel`) lub operatorem binarnym (`\mathbin`). Przykładowo, możemy mówić o relacji R i opisać, że elementy x i y są w relacji R, gdy `x \mathrel{R} y`

$$x \mathrel{R} y.$$

Porównajmy to z `x R y`

$$xRy.$$

Do definiowania własnych operatorów używamy komendy `\DeclareMathOperator` w preambule. Makro to ma dwa warianty. W pierwszym definiuje operatory funkcyjne, takie jak sin lub log,

```
\DeclareMathOperator{\Dom}{Dom}
```

w drugim definiuje operatory posiadające granice, podobne do lim

```
\DeclareMathOperator*{\argmax}{arg\,max}
```

[3] Patrz `http://tug.ctan.org/info/symbols/comprehensive/symbols-a4.pdf`.

[4] Patrz: `https://detexify.kirelabs.org/classify.html`.

[5] Pakiet `polski` dba o to, abyśmy otrzymali polskie znaki nierówności słabych z równoległymi kreskami, czyli $\leqslant$ i $\geqslant$ zamiast angielskich $\leq$ i $\geq$.

Operatory z granicami, takie jak lim lub zdefiniowany powyżej arg max, gdy umieścimy po nich indeks dolny, w trybie wyróżnionym umieszczają go pod operatorem. Na przykład

```
\begin{equation}
  \lim_{x\to\infty} f(x),
\end{equation}
```

da w wyniku

$$\lim_{x\to\infty} f(x),$$

podczas gdy `\(\lim_{x\to\infty} f(x)\)`, spowoduje pojawienie się wyrażenia w trybie liniowym $\lim_{x\to\infty} f(x)$. Zwróćmy uwagę, że podpis pod granicą w drugim przypadku został umieszczony nieco obok lim, aby nie powodować rozciągnięcia interlinii w pionie.

Podobne zachowanie mają inne duże operatory, do najpopularniejszych należy suma `\sum` oraz całka `\int`. Sumę składamy, podając jej górny i dolny indeks, na przykład

```
\begin{equation}
 \sum_{n=1}^{\infty} \frac{1}{n^2}.
\end{equation}
```

otrzymując

$$\sum_{n=1}^{\infty} \frac{1}{n^2},$$

W przypadku całki, musimy pamiętać o złożeniu operatora „d” krojem prostym

```
 \begin{equation}
  \int_{-\infty}^\infty \mathrm{e}^{-x^2} \operatorname{d}\!x.
 \end{equation}
```

otrzymując

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2}\,\mathrm{d}x.$$

Komenda `\!` użyta po definicji operatora „d”, kasuje niepotrzebnie dodany odstęp po operatorze jednoargumentowym. Gdybyśmy chcieli umieścić kilka wyrażeń obok siebie, przydatna jest komenda `\quad`, która dodaje wyraźny wizualnie odstęp.

Granice zwykle wyglądają dobrze, ale niestety są sytuacje, w których trzeba nad nimi popracować. Wyrażenie

$$g(c) = \sum_{a^2<b^2<c,b>1} f(a,b), \qquad \lim_{n\to\infty} \sup_{c^2\geqslant n} g(c)$$

to jedno z takich problematycznych definicji. Po pierwsze odstęp wokół znaku sumowania jest za duży, po drugie granice po prawej stronie nie są wyrównane w pionie. Komendą `\mathclap` możemy powiedzieć LaTeXowi, żeby granica nie zajmowała miejsca w poziomie. Komendą `\cramped` możemy wymusić nieco niższe potęgi (jak w pierwiastkach). Komenda `\crampedclap` łączy dwie powyższe. Komendy te biorą jeden argument będący parametrem.

Komenda `\adjustlimits` pozwala na wyrównanie granic w pionie. Komenda `\substack` działająca jak tablica z jedną kolumną pozwala umieścić wyrażenia w graniach jedno pod drugim. Stosując te triki, jesteśmy w stanie uzyskać o wiele precyzyjniejszy skład problematycznego wzoru

$$g(c) = \sum_{\substack{a^2<b^2<c\\b>1}} f(a,b), \qquad \lim_{n\to\infty}\ \sup_{c^2\ge n} g(c).$$

Zobaczmy, jak ich użyć w tym przypadku.

```
g(c) = \sum_{\crampedclap{\substack{a^2 < b^2 < c\\b>1}}} f(a, b), \quad
\adjustlimits \lim_{n\to\infty} \sup_{\cramped{c^2\ge n}} g(c).
```

W wyrażeniach szczególną uwagę powinniśmy poświęcić operacji mnożenia. W rozwinięciu dwumianu kwadratowego możemy napisać

$$2ab \quad 2\,ab \quad 2\cdot a\cdot b, \qquad 2\cdot ab \quad 2\cdot a\,b$$

i wiele więcej wariantów. Jedna ze standardowych zasad umieszczania znaków mnożenia, mówi:

1. umieszczamy najpierw wszystkie wartości liczbowe, potem symboliczne, potem uwikłane w wyrażeniach,
2. wartości liczbowe od wartości symbolicznych, oddzielamy przynajmniej odstępem `\,`,
3. wartości liczbowe oddzielamy od siebie znakiem mnożenia `\cdot`,
4. jeśli w wartościach uwikłanych pierwszym czynnikiem jest ułamek, blok wartości uwikłanych poprzedzamy znakiem mnożenia `\cdot`.

Zastosujmy tę zasadę do kilku przypadków.

- Zacznijmy od wcześniejszego $2\,ab$. Mamy blok liczbowy (liczba 2) oraz dwa czynniki w bloku symboli (stałe a oraz b). Dwójkę od stałych oddzielamy odstępem, ale nie ma potrzeby oddzielać od siebie symboli, napiszemy więc `2\,ab`.
- Jeśli mamy kilka liczb, symbol mnożenia jest konieczny — $1\cdot 2$ jest jednoznaczne, ale $1\,2$ jest bardzo podobne do 12.
- Jeśli pierwszym składnikiem wyrażenia uwikłanego jest ułamek, mnożenie jest konieczne, aby odróżnić $2\cdot\frac{1}{2}=1$ od $2\frac{1}{2}=2{,}5$ — w tym przypadku sam odstęp również nie wystarczy.

Znak × postarajmy się zachować dla iloczynu kartezjańskiego, iloczynu wektorowego i podawania wymiarów prostokątów. Znak $*$ nie powinien być używany w roli mnożenia, zachowajmy ten symbol dla splotu.

Często nasze wzory chcemy czymś otoczyć. Możemy otoczyć wzory pierwiastkiem kwadratowym `\sqrt{x}` lub n-tego stopnia `\sqrt[n]{x}`. Możemy też otoczyć wzory nawiasami. Niestety, jeśli napiszemy (`\frac{1}{2}`), nawiasy pozostaną niskie. Aby powiedzieć LaTeXowi, że nawiasy te mają się skalować tak, aby objąć całość wyrażenia, używamy komend `\left` i `\right`, które pozwalają wskazać lewy i prawy nawias.

```
\begin{equation}
 \left( \frac{1}{2} \right)
\end{equation}
```

daje w wyniku

$$\left(\frac{1}{2}\right).$$

Nawiasy mogą być różne. Do najpopularniejszych należą

Nawias	Komenda	Wygląd
okrągły	`()`	()
kwadratowy	`[]`	[]
klamrowy	`\{\}`	{}
prosty	`||`	||
trójkątny	`\langle\rangle`	$\langle\rangle$
sufit	`\lceil\rceil`	$\lceil\rceil$
podłoga	`\lfloor\rfloor`	$\lfloor\rfloor$

W połączeniu z komendami `\left` i `\right`, można również stworzyć niewidzialny nawias komendą `\left.` lub `\right.`. Przydaje się on, gdy chcemy coś zademonstrować, na przykład sposób znajdowania wartości ułamka łańcuchowego

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}.$$

Jeśli wprowadzimy nawiasy komendami

```
\begin{equation}
 x=1 + \cfrac{1}{\left.1+\cfrac{1}{1+\ldots}\right\} = x},
\end{equation}
```

otrzymamy

$$x = 1 + \cfrac{1}{\left.1 + \cfrac{1}{1 + \ldots}\right\} = x},$$

gdzie nawias klamrowy zamykający ma do pary niewidzialny nawias otwierający, dzięki czemu rozciąga się na całą wysokość mianownika. Po takiej ilustracji widać, że należy rozwiązać równanie

$$x = 1 + \frac{1}{x},$$

czyli

$$x^2 = x + 1,$$

co łatwiej rozwiązać, niż wyjściowy problem z ułamkiem łańcuchowym.

Niekiedy nawiasy są częścią wyrażenia. Tak jest na przykład w symbolu Newtona, który złożymy za pomocą komendy `\binom`

```
\binom{n}{k} = \frac{n!}{(n-k)!\,k!}
```

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}.$$

Oprócz poznanej strzałki `\to`, możemy użyć wielu rozciągliwych strzałek:

- `\mapsto`, $A \mapsto B$,
- `\rightarrow`, $A \rightarrow B$,
- `\Rightarrow`, $A \Rightarrow B$,
- `\leftarrow`, $A \leftarrow B$,
- `\Leftarrow`, $A \Leftarrow B$,
- `\leftrightarrow`, $A \leftrightarrow B$,
- `\Leftrightarrow`, $A \Leftrightarrow B$,
- `\xmapsto[x]{y}`, $A \xmapsto[x]{y} B$,
- `\xrightarrow[x]{y}`, $A \xrightarrow[x]{y} B$,
- `\xRightarrow[x]{y}`, $A \xRightarrow[x]{y} B$,
- `\xleftarrow[x]{y}`, $A \xleftarrow[x]{y} B$,
- `\xLeftarrow[x]{y}`, $A \xLeftarrow[x]{y} B$,
- `\xleftrightarrow[x]{y}`, $A \xleftrightarrow[x]{y} B$,
- `\xLeftrightarrow[x]{y}`, $A \xLeftrightarrow[x]{y} B$.

Pewien tekst możemy umieścić pod lub nad dowolnym operatorem, na przykład, aby umieścić literkę „H” nad równością, napiszemy `\overset{\mathrm{H}}{=}` uzyskując $\overset{\mathrm{H}}{=}$, który możemy wykorzystać, gdy równość wynika z reguły de l'Hospitala.

4 Składanie rozbudowanych wzorów

Niekiedy wyrażenia nie mieszczą się w linii. Wyrażenie na wiele linii składamy środowiskiem `multline` zamiast `equation`.

```
\begin{multline}
 a + b + c + d + e + f + g \\
 + h + i + j + k + l + m + n + o + p + q + r + s \\
 + t + u + v + w + x + y + z.
\end{multline}
```

$$\begin{multline}
a + b + c + d + e + f + g \\
+ h + i + j + k + l + m + n + o + p + q + r + s \\
+ t + u + v + w + x + y + z.
\end{multline}$$

Przejście do nowej linii powinniśmy umieścić tuż przed symbolem, przy czym od normy ISO 80000-2:2019 nie powinniśmy powtarzać symbolu przy przenoszeniu.

Jeśli składamy ciąg równań, najlepiej użyć środowiska `split*` wewnątrz `equation`.

```
\begin{equation}
 \begin{split}
  a + b
  & = c + d + e + f + g  \\
  & \quad + h + i + j  \\
  & \leq k.
 \end{split}
\end{equation}
```

$$\begin{equation}
\begin{split}
a + b & = c + d + e + f + g \\
& \quad + h + i + j \\
& \leq k.
\end{split}
\end{equation}$$

Zwróćmy uwagę, że wyrównywanie następuje na symbolu `&`, przy czym zaraz po tym symbolu występuje relacja, natomiast przenoszone operatory są przesuwane o `\quad` w prawo.

Do składu funkcji definiowanych przypadkami, najlepiej wewnątrz `equation` użyć `cases`.

```
\begin{equation}
 f(x) = \begin{cases}
  1, & \text{gdy } x > 0,\\
  0, & \text{gdy } x = 0,\\
  -1, & \text{gdy } x < 0.
 \end{cases}
\end{equation}
```

$$\begin{equation}
f(x) = \begin{cases}
1, & \text{gdy } x > 0,\\
0, & \text{gdy } x = 0,\\
-1, & \text{gdy } x < 0.
\end{cases}
\end{equation}$$

$$f(x) = \begin{cases} 1, & \text{gdy } x > 0, \\ 0, & \text{gdy } x = 0, \\ -1, & \text{gdy } x < 0. \end{cases}$$

Choć w tym przypadku przydatna będzie też komenda `\phantom` składająca niewidzialny tekst o zadanej szerokości.

```
\begin{equation}
 f(x) = \begin{cases}
  \phantom{-}1, & \text{gdy } x > 0,\\
  \phantom{-}0, & \text{gdy } x = 0,\\
  -1,           & \text{gdy } x < 0.
 \end{cases}
\end{equation}
```

$$f(x) = \begin{cases} 1, & \text{gdy } x > 0, \\ 0, & \text{gdy } x = 0, \\ -1, & \text{gdy } x < 0. \end{cases}$$

Jeśli równania mają być wyrównane, możemy użyć środowiska `aligned` wewnątrz `\equation`.

```
\begin{equation}
 \begin{aligned}
 x &= y & y&= z+2 \\
 z+3 &= y+1 & x-2 &= z
 \end{aligned}
\end{equation}
```

$$\begin{aligned} x &= y & y&= z+2 \\ z+3 &= y+1 & x-2 &= z \end{aligned}$$

Jeśli równania mają być tylko wpisane (bez wyrównywania), możemy użyć środowiska `gathered` wewnątrz `\equation`.

```
\begin{equation}
 \begin{gathered}
  x = y + y + z \\
  a + b + c + d + e + f = g
 \end{gathered}
\end{equation}
```

$$\begin{gathered} x = y + y + z \\ a + b + c + d + e + f = g \end{gathered}$$

Do składu macierzy służy zbiór środowisk `matrix`. Z literą `p` są w nawiasach okrągłych, `b` prostokątnych, `B` klamrowych, `v` kreskach pionowych pojedynczych i `V` podwójnych.

```
A_{m,n} =  \begin{pmatrix}
 a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
 a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
 \vdots  & \vdots  & \ddots & \vdots  \\
 a_{m,1} & a_{m,2} & \cdots & a_{m,n}
\end{pmatrix}.
```

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}.$$

Domyślnie, komendy macierzy wyrównują kolumny do środka. Wariant komend z gwiazdką posiada dodatkową opcję do wyrównywania.

```
\begin{bmatrix*}[r]
 100 & 1 \\
 0 & 50
\end{bmatrix*}.
```

$$\begin{bmatrix*}[r] 100 & 1 \\ 0 & 50 \end{bmatrix*}.$$

Powinniśmy unikać ręcznego robienia macierzy z tabelek, ponieważ są problemy z odstępami pomiędzy nawiasami a komórkami.

```
 \begin{Bmatrix}
 1 & 2 \\
 3 & 4
\end{Bmatrix}, \left\{\begin{array}{cc}
 1 & 2 \\
 3 & 4
\end{array}\right\}
```

$$\begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}, \left\{\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}\right\}$$

5 Twierdzenia i definicje

Pakiet `amsthm` dostarcza komendy do budowy twierdzeń, definicji oraz innych często spotykanych w pracach matematycznych wyróżnionych fragmentów tekstu. Za pomocą komendy `\newtheorem` w preambule możemy definiować nowe środowiska twierdzeń, przykładowo:

```
\newtheorem{tw}{Twierdzenie}[chapter]
\newtheorem{lem}[tw]{Lemat}
```

co spowoduje stworzenie środowiska `tw` numerującego twierdzenia od 1 dla każdego z rozdziałów. Dodatkowo w tej samej numeracji znajdują się lematy (środowisko `lem`). W obu przypadkach, napisy „Twierdzenie” oraz „Lemat” będą dzięki temu w języku polskim. Pakiet `amsthm` definiuje też środowisko `proof` do dowodów z komendą `\qedhere`, wstawiającą kwadrat oznaczający koniec dowodu. Możemy zapisać

```
\begin{tw}[Nazwisko]
 Treść bardzo trudnego twierdzenia.
\end{tw}
\begin{proof}
 Ćwiczenie. \qedhere
\end{proof}
```

co daje nam

Twierdzenie 1 (Nazwisko)**.** *Treść bardzo trudnego twierdzenia.*

Dowód. Proste ćwiczenie. □