

Technologie Informacyjne - Laboratorium

L^AT_EX- pakiet tikz

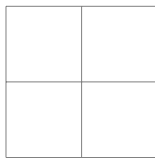
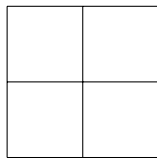
prowadzący: M. Emirsajłow, A. Gnatowski, R. Idzikowski, T. Niżyński

1 Podstawowe kształty


Na poprzednich zajęciach zapoznaliśmy się z pakietem `tikz` oraz `pgfplots`, których używaliśmy do rysowania wykresów. Celem dzisiejszego laboratorium jest poszerzenie wiedzy na temat pakietu `tikz`. Ponownie będziemy używać otoczenia `tikzpicture`. Do rysowania różnych obiektów będziemy używać rozkazu:

`\draw[opcje] (parametr_1) kształt (parametr_2),`

gdzie jako `parametr_1` oraz `parametr_2` będziemy podawać zazwyczaj współrzędne, między którymi ma być rozciągnięty obiekt. Jeśli chcemy narysować prostokąt lub siatkę wystarczy podać współrzędne przeciwległych punktów. Współrzędne początkowe możemy wprowadzić dowolnie.

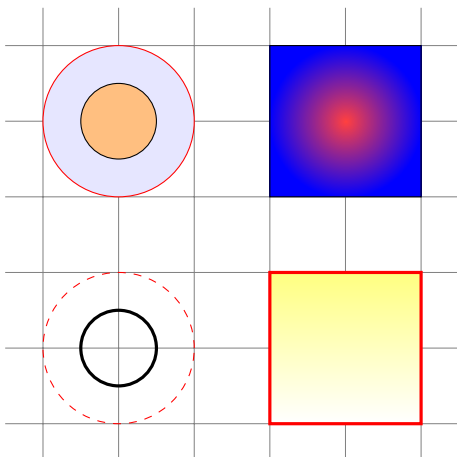


```
1 \centering
2 \begin{tikzpicture}
3   \draw (-1,-1) grid (1,1);
4   \draw[help lines] (2,-1) grid (4,1);
5 \end{tikzpicture}
```

Możemy rysować pojedyncze kształty bez zastosowania otoczenia `tikzpicture`, należy wtedy poprzedzić polecenie `\draw` rozkazem `tikz`, np.: `\tikz\draw[fill=orange] (0,0) circle (1ex);` a w efekcie otrzymamy . W pakiecie mamy też dostępne inne kształty:








`grid` siatka
`rectangle` prostokąt
`circle` okrąg, jako drugi parametr należy podać promień okręgu
`ellipse` elipsa, jako drugi parametr należy podać promień w poziomie i pionie
`--` linia, możliwe opcje `->` (strzałka), `<->` (dwustronna strzałka)

Podobnie jak na poprzednim laboratorium możemy zmieniać różne parametry kształtów, ponownie możemy dodać opcje `dashed` lub `dotted`, aby kształt był wyrysowany linią przerywaną lub kropkowaną. Opcja `color` dotyczy się linii figury, jeśli chcemy wypełnić jednolitym kolorem należy użyć opcji `fill`. Możemy również używać opcji cieniowania: (1) polecenia `top color`, `middle color`, `bottom color` pozwolą cieniować względem poziomych osi, (2) polecenia `inner color` i `outer color` służą co cieniowa promieniowego.

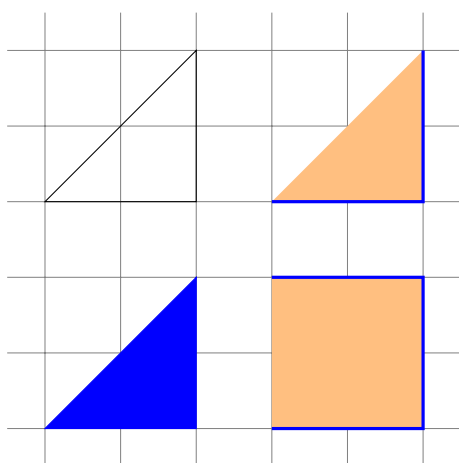


```
1 \centering
2 \begin{tikzpicture}
3   \draw[help lines] (-0.5,-3.5) grid (5.5,2.5);
4   \draw[red,fill=blue!10] (1,1) circle (1cm);
5   \draw[red,dashed] (1,-2) circle (1cm);
6   \draw[very thick] (1,-2) circle (0.5cm);
7   \draw[fill=orange!50] (1,1) circle (0.5cm);
8   \draw[outer color = blue, inner color = red!75 ]
9     (3, 0) rectangle (5,2) ;
10   \draw[red,very thick,top color=yellow!50]
11     (3,-3) rectangle (5,-1);
12 \end{tikzpicture}
```

Do rysowania kształtów możemy stosować linie o zadanej szerokości przy użyciu parametru `line width` lub skorzystać z kilku zdefiniowanych:

| | | |
|--------------------------|---|------------------------|
| <code>ultra thin</code> |  | linia o grubości 0.1pt |
| <code>very thin</code> |  | linia o grubości 0.2pt |
| <code>thin</code> |  | linia o grubości 0.4pt |
| <code>semithick</code> |  | linia o grubości 0.6pt |
| <code>thick</code> |  | linia o grubości 0.8pt |
| <code>very thick</code> |  | linia o grubości 1.2pt |
| <code>ultra thick</code> |  | linia o grubości 1.6pt |

Przy użyciu zwykłej linii `--` możemy tworzyć własne kształty na podstawie podanych punktów. W takim wypadku możemy tworzyć ciąg złożonych z kilku punktów. W celu domknięcia cyklu możemy użyć polecenia `cycle` lub podać wprost pierwszy punkt. Tak powstałe kształty możemy również wypełniać kolorem. Rozkaz `\filldraw` działa bardzo podobnie do swojego odpowiednika `draw`, tylko wypełnia cały kształt kolorem.

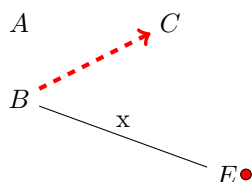


```

1 \centering
2 \begin{tikzpicture}
3   \draw[help lines] (-0.5,-3.5) grid (5.5,2.5);
4   \draw (0,0) -- (2,0) -- (2,2) -- cycle;
5   \filldraw[fill=blue]
6     (0,-3) -- (2,-3) -- (2,-1) -- cycle;
7   \draw[fill=orange, very thick]
8     (3,0) -- (5,0) -- (5,2) -- cycle;
9   \draw[fill=orange, very thick]
10    (3,-3) -- (5,-3) -- (5,-1) -- (3,-1) -- cycle;
11 \end{tikzpicture}

```

W pakiecie `tikz` niezbędnym oraz bardzo wszechstronnym elementem są węzły, pomagają między innymi odwoływać się wielokrotnie do tych samych elementów oraz używać do nich różnych stylów. Węzły definiujemy przy użyciu polecenia `\node (nazwa){etykieta}`, gdzie musimy podać jego nazwę oraz tekst do wyświetlenia, który może pozostać pusty. Możemy wymusić położenie węzła w konkretnym miejscu `\node (nazwa) at (współrzędne){etykieta}`; . Kolejne węzły mogą być umieszczane w konkretnych współrzędnych lub względem innych węzłów poprzez opcję `below of=nazwa`. Odległością między węzłami możemy sterować za pomocą opcji `node distance` dla otoczenia `tikzpicture`. Węzły możemy łączyć za pomocą kształtów. W celu sformatowania węzła jako typowy punkt, należy narysować wypełniony okrąg o małej średnicy. Możemy łączyć polecenia tworzenia węzła po narysowaniu obiektu. Należy jeszcze potem przy pomocy opcji przesunąć etykietę, ponieważ domyślnie tekst jest wyświetlany na środku współrzędnych, możemy zrobić to za pomocą opcji `above right` (lub `below`, `left` itp.) albo przesunąć ręcznie o zadaną długość rozkazami `xshift` oraz `yshift` podając o ile chcemy przesunąć daną etykietę. Możemy utworzyć również węzeł na linii między dwoma punktami używając opcji `midway`.



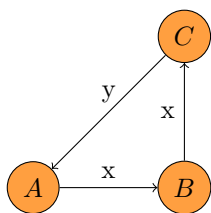
```

1 \centering
2 \begin{tikzpicture}[node distance = 1cm]
3   \node (A) at (0,1) {$A$};
4   \node [below of=A] (B) {$B$};
5   \node (C) at (2,1) {$C$};
6   \draw [red,dashed, ultra thick, ->] (B) -- (C);
7   \draw [fill=orange] (4,0.5) circle (2pt)
8     node (D)[above right] {$D$};
9   \draw [fill=red] (3,-1) circle (2pt)
10    node (E)[xshift=-0.25cm] {$E$};
11   \draw (B) -- (E) node[midway, above] {x};
12 \end{tikzpicture}

```

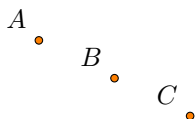
2 Definiowanie stylów

Przy rysowaniu grafów, schematów często będziemy spotykali elementy, które wyglądają podobnie. W takim wypadku warto definiować style, żeby nie trzeba było opisywać za każdym razem powtarzalnych elementów. Style definiujemy w opcjach do otoczenia `tikzpicture`. Format definiowanego stylu jest następujący `nazwa/.style={ustawienia}`. Jeśli chcemy zrobić graf to każdy węzeł grafu ma taką samą strukturę, ale różni się zawartością (w tym wypadku etykietą). Musimy wybrać odpowiedni kształt, np.: `circle` – koło (domyślnie jest kwadrat). Jednak to nie wszystko, aby wyrysować kształt należy wymusić rysowanie opcją `draw`. Opcja `color` zmienia nam kolor obramowania oraz czcionki, jeśli chcemy zmienić jedynie kolor wypełnienia należy ustawić opcję `fill` na odpowiedni kolor. Oczywiście można korzystać ze wszystkich wcześniej poznanych elementów jak między innymi strzałki.



```
1 \centering
2 \begin{tikzpicture}[
3   vertex/.style={circle, draw, fill=orange!75}
4 ]
5   \node (A)[vertex] at (0,0) {$A$};
6   \node (B)[vertex] at (2,0) {$B$};
7   \node (C)[vertex] at (2,2) {$C$};
8   \draw[>-] (A) -- (B) node[midway, above] {x};
9   \draw[>-] (B) -- (C) node[midway, left] {x};
10  \draw[>-] (C) -- (A) node[midway, above] {y};
11 \end{tikzpicture}
```

Jeśli chcemy zdefiniować styl dla punktów postaci P . Będziemy musieli zdefiniować również osobny styl dla etykiety, który nazwiemy `mylabel`. Użyjemy standardowego polecenia do etykiet `label=pozycja:etykieta`, ale ze zmienionym położeniem `above left`, jako treść etykiety podamy tekst przekazany jako parametr pierwszy `#1` przy wywołaniu stylu `mylabel`. Dla przejrzystego efektu wyświetlimy treść w trybie matematycznym `$$`. Na koniec musimy wymusić jeszcze zmniejszenie węzła, ponieważ mimo braku tekstu wewnątrz pozostaje pusta przestrzeń (dzięki niej tekst w standardowym węźle nie pokrywa się z obrysem węzła). Możemy zmienić wartość pustej przestrzeni poprzez zmianę opcji `inner sep=wymiar`. Teraz wystarczy wywołać węzeł ze stylem `vertex` oraz jako parametr wpisać naszą etykietą pozostawiając puste nawiasy klamrowe.



```
1 \centering
2 \begin{tikzpicture}[
3   vertex/.style=
4     {circle, draw, fill=orange, inner sep=1pt},
5   mylabel/.style = {label={above left:{$#1$}}}
6 ]
7   \node (A)[vertex, mylabel = A] at (0,1.5) {};
8   \node (B)[vertex, mylabel = B] at (1,1) {};
9   \node (C)[vertex, mylabel = C] at (2,0.5) {};
10 \end{tikzpicture}
```

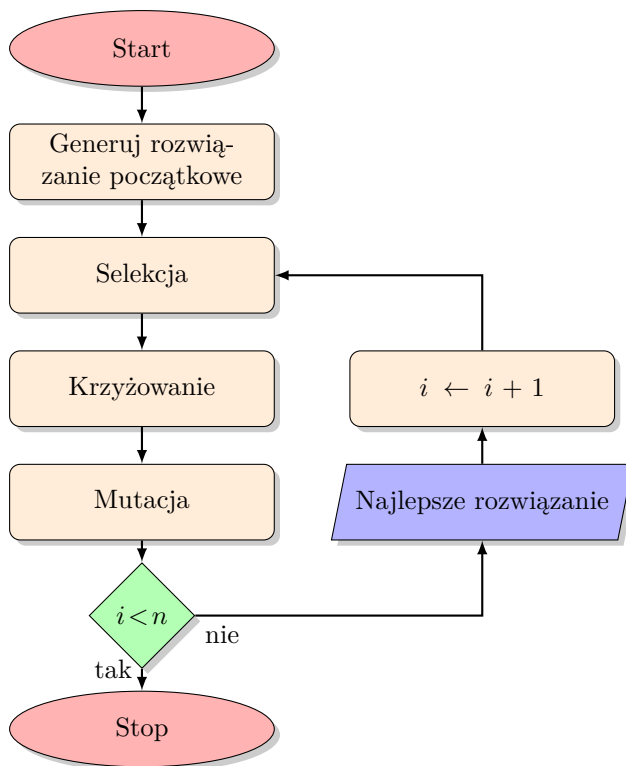
Między innymi podczas kursu podstaw programowania można się spotkać z schematami blokowymi. Warto nauczyć się robić je przy użyciu pakietu `tikz`. Zaczniemy od zdefiniowania stylu bazowego po którym inne będą dziedziczyć, ponieważ wszystkich blozków będą dotyczyć parametry takie jak: minimalna szerokość bločka `minimum width`, minimalna wysokość `minimum height`, formatowanie tekstu `text centered`, wielkość czcionki `font` i jej kolor, dodatkami będzie ustawienie cienia pod każdym bločkem `drop shadow=black!40`, do tego potrzebujemy dołączyć cienie z biblioteki elementów `tikz` `\usetikzlibrary{shadows}` (Proszę skopiować raport z linku referencyjnego, ponieważ zostały tam dodane cienie, dodatkowe kształty i nietypowe strzałki). Następnie musimy zdefiniować 4 typu blozków, których będziemy używać (graniczny, wejścia/wyjścia, operacyjny i decyzyjny). Bločki będą różnić się głównie kształtem i kolorem, jednak różne kształty przyjmują różne parametry. Dla trapezu wystarczy podać dwa kąty `trapezium left angle` oraz `trapezium right angle` oraz włączyć opcję `trapezium stretches=true`, która odpowiednio rozciągnie trapez do zadanej szerokości i kątów. Dla prostokątów możemy włączyć zaokrąglone rogi `rounded corners`. Jako że bloki często mają dłuższy tekst warto wymusić załamanie wierszy poprzez ograniczenie szerokości tekstu `text width=3cm`. Dla diamentu warto przeddefiniować minimalną szerokość na mniejszą, aby nie był zbyt duży.

Możemy również zdefiniować styl dla strzałek. Warto je pogrubić ustawieniem **thick**, aby był czytelniejszy. Strzałki **-latex** są ładną alternatywą dla standardowych strzałek **->**. Bloczki możemy tak jak inne węzły umieszczać poprzez wskazywanie konkretnych współrzędnych lub poprzez wskazanie orientacji względem innych węzłów. Wtedy odstępem między nimi możemy sterować za pomocą opcji **node distance**. Teraz wystarczy tylko przy odpowiednich węzłach wskazać konkretny styl i wpisać etykietę. Pionowe lub poziome strzałki (ogólnie w linii prostej) robimy standardowo, jednak gdy chcemy zrobić strzałkę z kątem prostym to musimy wskazać kolejno orientacje strzałki przed i po załamaniu: pozioma - czy pionowa **|**. Opcja **at start**, **at end** lub **midway** wskaże gdzie ma się znaleźć etykieta opisująca strzałkę. Warto też wskazać ułożenie etykiety przy pomocy opcji **anchor=kierunek**.

```

1 \begin{tikzpicture}[
2   baseshape/.style={minimum width=3.5cm, minimum height=1cm,
3     text centered, font=\normalsize,
4     draw=black, drop shadow=black!40},
5   startstop/.style={baseshape, ellipse, fill=red!30},
6   io/.style={baseshape, trapezium, trapezium stretches=true,
7     trapezium left angle=70, trapezium right angle=110, fill=blue!30},
8   process/.style={baseshape, rectangle, rounded corners, text width=3cm, fill=orange!15},
9   decision/.style={baseshape, diamond, minimum width=1cm, fill=green!30},
10  arrow/.style={thick, -latex},
11  node distance=1.5cm,
12 ]
13 \node (step1) [startstop] {Start};
14 \node (step2) [process, below of=step1] {Generuj rozwi\k{a}zanie pocz\k{a}tkowe};
15 \node (step3) [process, below of=step2] {Selekcja};
16 \node (step4) [process, below of=step3] {Krzy\.{z}owanie};
17 \node (step5) [process, below of=step4] {Mutacja};
18 \node (step7) [decision, below of=step5] {$i \leq n$};
19 \node (step8) [process, right of=step4, xshift=3cm] {$i \leftarrow i + 1$};
20 \node (step9) [io, below of=step8] {Najlepsze rozwi\k{a}zanie};
21 \node (step10) [startstop, below of=step7] {Stop};
22 \draw[arrow] (step1) -- (step2);
23 \draw[arrow] (step2) -- (step3);
24 \draw[arrow] (step3) -- (step4);
25 \draw[arrow] (step4) -- (step5);
26 \draw[arrow] (step5) -- (step7);
27
28 \draw[arrow] (step7) -- node[at start, anchor=east] {tak} (step10);
29 \draw[arrow] (step7) -- node[at start, anchor=north west] {nie} (step8);
30 \draw[arrow] (step8) -- (step3);
31 \draw[arrow] (step9) -- (step8);
32 \end{tikzpicture}

```



3 Pętle

Podobnie jak w programowaniu, także przy tworzeniu wykresów i schematów przydatne okazują się pętle oraz instrukcje warunkowe. W tej sekcji pokrótce omówiona zostanie pętla `foreach` pochodząca z paczki `pgfplots`. Pętla ta ma bardzo elastyczną składnię, umożliwiając automatyzowanie wielu czynności:

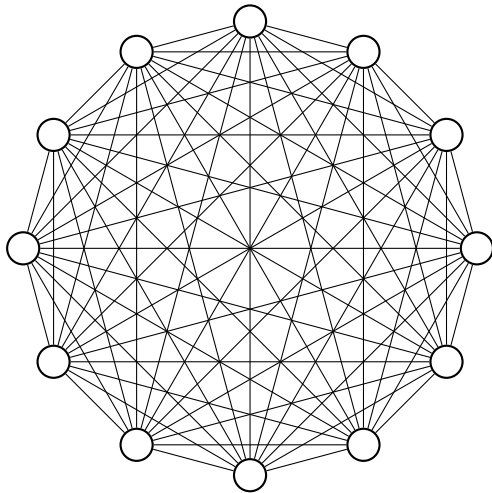
```
\foreach \iterator [opcje] in {zakres}{
```

```

    zawartość pętli
}

```

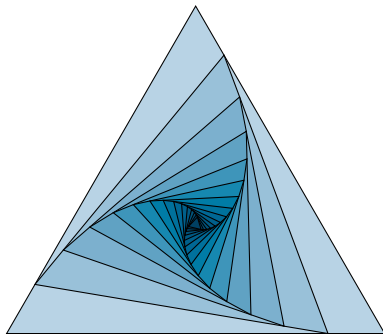
Podstawowym elementem jest zmienna `\iterator`, której wartość zmienia się w każdej iteracji. Zakres pętli definiowany jest w intuicyjny sposób, np.: `{1,...,5}` generuje zakres od 1 do 5, co 1. Więcej przykładów można znaleźć w dokumentacji oraz Internecie.



```

1 \begin{tikzpicture}
2   \def\n{12}
3   \foreach \i in {1,...,\n}{%
4     \pgfmathparse{(\i-1)*(360/\n)}
5     \node[draw, circle, inner sep=0.15cm]
6       (n\i) at (\pgfmathresult:3cm) [thick] {};
7   }
8   \foreach \i [count=\ii from 1] in {2,...,\n}
9     \foreach \j in {\i,...,\n}
10      \path (n\ii) edge[-] (n\j);
11 \end{tikzpicture}

```



```

1 % Creator: Martin Scharrer
2 \begin{tikzpicture}
3   \coordinate (A) at (0,0);
4   \coordinate (B) at (-60:5cm);
5   \coordinate (C) at (240:5cm);
6   \foreach \density in {20,30,...,160}{%
7     \draw[fill=MidnightBlue!\density]
8       (A)--(B)--(C)--cycle;
9     \path
10      (A) coordinate (X)
11      -- (B) coordinate[pos=.15] (A)
12      -- (C) coordinate[pos=.15] (B)
13      -- (X) coordinate[pos=.15] (C);
14   }
15 \end{tikzpicture}

```