Ćwiczenie 9
Typst i Touying

9.1. Rozpoczęcie pracy

1. Uruchomić Visual Studio Code i terminal. Przygotować katalog roboczy do edycji dokumentu Typst i prezentacji Touying. Sprawdzić dostępność kompilatora:

```
typst --version
typst help
```

   Wymagany jest **Typst >=0.15.0**. Jeżeli program nie jest zainstalowany, pobrać oficjalne wydanie z `https://github.com/typst/typst/releases` albo skorzystać z `https://typst.app/`.

2. W VS Code można zainstalować rozszerzenie *Tinymist Typst* zapewniające podgląd. W aplikacji webowej utworzyć nowy projekt i zapoznać się z eksportem źródeł oraz PDF. Pierwsze użycie pakietu Touying w lokalnym kompilatorze wymaga pobrania zależności lub wcześniej przygotowanej pamięci podręcznej pakietów.

3. Pobrać trzy pliki z kanału zajęć albo bezpośrednio z załączników niniejszego pliku PDF: notatka.typ, makra.typ, slajdy.typ. Jeżeli czytnik pdf nie pokazuje załączników, użyć udostępnionego przez prowadzącego archiwum zip.
   Umieścić je we wspólnym katalogu `~/cw9`: `notatka.typ`, `makra.typ` i `slajdy.typ`. Sprawdzić, czy znajdują się obok siebie:

```
mkdir -p ~/cw9
cd ~/cw9
ls -la
```

9.2. Pierwszy dokument Typst

4. Otworzyć `notatka.typ`. Porównać nagłówki `= Tytuł` i `== Sekcja` z poleceniami struktury dokumentu w LaTeXu.
   Zidentyfikować znaczniki pogrubienia, kursywy, list oraz tekstu źródłowego.
   Sprawdzić rolę znaku `#` (będzie potrzebne później).

5. Wykonać kompilację i uruchomić automatyczne odświeżanie:

```
typst compile notatka.typ notatka.pdf
typst watch notatka.typ notatka.pdf
```

   Zmienić jedno zdanie, sprawdzić wynik w PDF i zatrzymać `watch` przez `Ctrl+C`.
   W aplikacji webowej obserwować podgląd na żywo.

6. Ustawić format strony, tekst, język i numerację sekcji:

```
#set page(paper: "a4", margin: 22mm)
#set text(lang: "pl", size: 11pt)
#set heading(numbering: "1.")
```

   Wpisać własny identyfikator, tytuł i datę. Porównać reguły `#set` z konfiguracją preambuły LaTeXa.

7. Dodać rozdział o skrypcie `deploy.sh` z ćwiczenia 7: dane wejściowe, transmisja przez SSH i SCP, przetwarzanie Bash oraz wynik `Liczba_wierszy=4`. Nie publikować danych rzeczywistego serwera, loginów, tokenów ani prywatnych kluczy.

8. Wprowadzić równanie i tabelę, korzystając z poniższej składni:

```
$ sum_(i=1)^n i = (n(n+1)) / 2 $

#table(
  columns: (1fr, 2fr),
  table.header([*Etap*], [*Wynik*]),
  [Test], [Liczba_wierszy=4],
  [Wdrożenie], [Plik wynikowy],
)
```

   Porównać je ze środowiskami `equation` i `tabular` w LaTeXu.

9. Dodać podpisany element graficzny bez zewnętrznych obrazów:

```
#figure(
  rect(width: 70\%, height: 12mm, fill: luma(220)),
  caption: [Schemat zdalnego przetwarzania],
)
```

   Sprawdzić numerację i podpis. W razie dodania grafiki zewnętrznej podać jej źródło oraz upewnić się, że można ją wykorzystać.

9.3. Własne funkcje zamiast makr

10. Otworzyć `makra.typ`. Przeanalizować definicje funkcji przez `#let`, parametry pozycyjne i nazwane wraz z wartościami domyślnymi. Porównać je z makrami tworzonymi przez `\newcommand` w ćwiczeniach 4 i 5.

11. W `notatka.typ` wykorzystać import:

```
#import "makra.typ": wyroznij, ramka
```

   Użyć funkcji `wyroznij` trzykrotnie, w tym raz ze zmienionym kolorem, oraz funkcji `ramka` do wyróżnienia oczekiwanego wyniku.

12. Samodzielnie dopisać do `makra.typ` funkcję `krok(numer, opis)`. Powinna przyjmować dwa argumenty i jednolicie formatować kolejne etapy przepływu SSH – SCP – Bash – SCP. Zaimportować ją do dokumentu i zastosować co najmniej trzy razy.

13. Wprowadzić własny styl lub funkcję, której pojedyncza zmiana wpłynie na kilka elementów. Opisać różnice między `#let`, `#set` i `#show`. Wskazać analogiczne zastosowanie makr, pakietów i definicji środowisk LaTeXa.

14. Celowo wprowadzić błąd składni, odczytać komunikat kompilatora, naprawić dokument i sprawdzić kod zakończenia polecenia. Nie usuwać komunikatów błędów bez ustalenia ich przyczyny.

9.4. Pierwsze slajdy w Touying

15. Otworzyć `slajdy.typ`. Touying jest pakietem do tworzenia prezentacji w Typst, podobnym funkcjonalnie do *Beamera*. Wymagana para wersji to **Typst 0.15+ oraz Touying 0.8.0**:

```
#import "@preview/touying:0.8.0": *
#import themes.simple: *
#show: simple-theme.with(aspect-ratio: "16-9")
```

    Przy pierwszym użyciu kompilator może pobrać Touying z rejestru pakietów.

16. Skompilować prezentację:

```
typst compile slajdy.typ slajdy.pdf
```

    Porównać znaczenie `= Sekcja`, `== Slajd` i `#pause`. Dlaczego liczba stron w PDF może być większa niż liczba slajdów logicznych?

17. Przygotować co najmniej pięć **slajdów treści** (nie licząc tytułowego) o tym samym projekcie SSH; jeden slajd może obejmować dwa zagadnienia: cel, przepływ danych, sposób przetwarzania, funkcje Typst, wyniki i bezpieczeństwo. Utrzymać proporcje 16:9 oraz czytelne, krótkie komunikaty zamiast całych akapitów.

18. Użyć `#pause` na slajdzie opisującym kolejne etapy. Otworzyć wynikowy PDF i przejrzeć kolejne strony. Opisać różnicę między krokowym ujawnianiem treści, a filmową animacją.

19. Zastosować konstrukcję `#slide[...]` lub `#focus-slide[...]` motywu *Simple*. Skonfigurować wspólny element stylu prezentacji, np. stopkę (zob. dokumentacja motywu Simple), zamiast formatować każdy slajd ręcznie.

20. Umieścić w prezentacji tabelę wyników albo fragment kodu Typst. Porównać sposób zapisu z analogicznym elementem w dokumencie i sprawdzić, czy treść pozostaje czytelna w widoku pełnoekranowym.

9.5. Weryfikacja i zapis w Git

21. Utworzyć katalog wynikowy i ponownie skompilować pliki:

```
mkdir -p wyniki
typst compile notatka.typ wyniki/notatka.pdf
typst compile slajdy.typ wyniki/slajdy.pdf
```

    Sprawdzić obecność dwóch PDF, liczbę stron i czytelność równań.

22. Umieścić projekt we własnym repozytorium Git z ćwiczenia 8 albo utworzyć lokalne repozytorium. W `README.md` wskazać dokładne polecenia kompilacji, wymagane wersje oraz listę źródeł. Nie uruchamiać płatnych workflowów ani budowania prezentacji na uczelnianym serwerze.

23. Przygotować w `README.md` zwięzłe porównanie co najmniej jednej konstrukcji formatowania i jednej funkcji: osobno w LaTeXu (ćwiczenia 4–5) i Typst. Wyjaśnić, które elementy można przenosić między dokumentami i co pełni rolę szablonu.
24. Sprawdzić stan repozytorium, a następnie zapisać kod źródłowy, README i wybrane PDF w Git:

```
git status
git add README.md *.typ wyniki/*.pdf
git commit -m "Add Typst document and Touying slides"
```

Jeżeli katalog **nie jest jeszcze repozytorium Git**, wykonać `git init -b main`; w sklonowanym repozytorium nie inicjalizować Git ponownie. Przed `git add` upewnić się, że istnieje katalog `wyniki/` i oba pliki PDF. Wysyłać wyłącznie do własnego zdalnego repozytorium, bez danych uwierzytelniających.

9.6. Zadania dodatkowe

25. **Szablon wielokrotnego użytku.** Utworzyć funkcję `szablon(dokument)`, która konfiguruje stronę, krój pisma i nagłówki. Zastosować ją w dwóch dokumentach za pomocą `#import` i `#show`. Porównać efekt z plikiem makr lub stylem LaTeXa.
26. **Drugi motyw Touying.** Utworzyć wariant prezentacji z motywem *Metropolis* lub *University*. Wygenerować oddzielny PDF i porównać czytelność, nagłówki, numerację oraz liczbę podslajdów. Nie zmieniać treści każdego slajdu ręcznie.

9.7. Dokumentacja i wskazówki

Aktualne materiały: `https://typst.app/docs/`, `https://typst.app/docs/guides/for-latex-users/`, `https://typst.app/docs/tutorial/making-a-template/`, `https://typst.app/universe/package/touying/` i `https://touying-typ.github.io/`. W razie błędów sprawdzić wersję kompilatora, ścieżki importów i dostępność pakietu Touying. Nie zastępować źródeł zrzutami ekranu.

9.8. Zakończenie ćwiczenia

27. Przedstawić prowadzącemu `notatka.typ`, `makra.typ`, `slajdy.typ`, `README.md` i dwa PDF. Weryfikacja obejmuje poprawną kompilację, własną funkcję z argumentami, przynajmniej pięć slajdów treści, krokowe ujawnianie, czytelność oraz porównanie z LaTeXem.