Wrocław 2018
Cezary Hołub

Wrocławska Wyższa Szkoła Informatyki Stosowanej

Przedmiot	Zaawansowane Praktyki Programistyczne (Wykład + ćwiczenia)
Semestr	Lato 2018

Materiały do ćwiczeń - część 2

Lista nie jest na ocenę, jednak proszę się przyłożyć, gdyż następna lista z Git'a już będzie na ocenę.

System kontroli wersji Git

Konfiguracja i pierwsze kroki.

1. Utwórz nowy katalog w dowolnym miejscu na dysku. Przykładowa lokalizacja: c:/Users/Student/Pulpit/git-cwiczenia
2. Otwórz okno terminala. W przypadku systemu Windows będzie to aplikacja "Gitbash" (dostępna po zainstalowaniu Gita z http://git-scm.com/download). Przejdź do wcześniej stworzonego katalogu używając komendy "cd c:/Users/Student/Pulpit/git-cwiczenia" (ścieżka ma oczywiście odpowiadać lokalizacji utworzonego wcześniej folderu).
3. W katalogu utwórz nowe repozytorium Gita używając komendy "git init":

```
c:\dev\git-cwiczenia>git init
Initialized empty Git repository in c:/dev/git-cwiczenia/.git/

c:\dev\git-cwiczenia>
```

4. Repozytorium jest gotowe do użycia. W katalogu roboczym nie ma żadnych plików i co za tym idzie żadnych zmian które można zapisać (commit) w repozytorium.Zweryfikuj to używając komendy "git status":

```
c:\dev\git-cwiczenia>git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
```

5. Dodaj do katalogu roboczego nowy plik tekstowy "plik_1.txt" z poniższą zawartością:

Lorem ipsum dolor sit amet

6. Sprawdź status repozytorium komendą "git status"

```
c:\dev\git-cwiczenia>git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        plik_1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Z odpowiedzi programu możemy dowiedzieć się, że plik_1.txt nie jest zaznaczony do commita, tzn. nie jest dodany do staging area.

7. Spróbuj zapisać aktualny stan katalogu roboczego komendą “git commit”:

```
c:\dev\git-cwiczenia>git commit

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'cezary@cezary-VAIO.(none)')
```

Operacja nie udała się, należy skonfigurować konto.

8. Skonfiguruj nową nazwę autora i email:

```
git config user.name NAZWA_UZYTKOWNIKA
git config user.email EMAIL_UZYTKOWNIKA
```

Powyższe komendy dokonają konfiguracji tylko na poziomie używanego repozytorium. Znaczy to, że dla nowego repozytorium (w innym katalogu) użyte będą wartości domyślne. Aby skonfigurować Gita na poziomie globalnym, należy dodać flagę global, tzn:

```
git config --global user.name NAZWA_UZYTKOWNIKA
git config --global user.email EMAIL_UZYTKOWNIKA
```

Podajemy przykładowe dane:

```
c:\dev\git-cwiczenia>git config --global user.email "cezary.holub@cohesiva.com"

c:\dev\git-cwiczenia>git config --global user.name "cezary holub"
```

9. Ponownie zapisujemy aktualny stan katalogu roboczego komendą “git commit”;

```
c:\dev\git-cwiczenia>git commit
On branch master

Initial commit

Untracked files:
        plik_1.txt

nothing added to commit but untracked files present
```

Operacja nie powiodła się, ponieważ plik nie został dodany do staging area.

10. Dodaj plik do staging area:

```
c:\dev\git-cwiczenia>git add plik_1.txt
```

11. Sprawdź status. Odpowiedź programu potwierdzi gotowość do wykonania commita:

```
c:\dev\git-cwiczenia>git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   plik_1.txt
```

12. Wykonaj commit. Opis należy podać w podwójnych cudzysłowach za flagą m, przykładowo:

```
c:\dev\git-cwiczenia>git commit -m "dodano plik_1.txt"
[master (root-commit) a59e5e4] dodano plik_1.txt
 1 file changed, 1 insertion(+)
 create mode 100644 plik_1.txt
```

13. Sprawdź status katalogu roboczego komendą “git status”. Jeżeli commit został wykonany poprawnie, to odpowiedź programu powinna wskazywać na brak niezapisanych zmian w katalogu roboczym (“working directory clean”).

```
c:\dev\git-cwiczenia>git status
On branch master
nothing to commit, working directory clean
```

14. Sprawdź historię repozytorium używając komendy “git log”:

```
c:\dev\git-cwiczenia>git log
commit a59e5e44f0bd2681162120d031f42b29484fa2e4
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:20:45 2015 +0100

    dodano plik_1.txt
```

Z odpowiedzi programu dowiedzieć się możemy następujących informacji:

- W repozytorium zapisany jest jeden commit
- Każdy commit identyfikowany jest sumą kontrolną SHA1, w tym przypadku jest to a59e5e44f0bd2681162120d031f42b29484fa2e4
- Opis commita jest zgodny z argumentem który użyliśmy wcześniej z komendą “git commit”
- Git zapisał datę oraz autora danego commita. Dane autora zostały skonfigurowane automatycznie na podstawie nazwy zalogowanego użytkownika i nazwy komputera

15. Zmień zawartość pliku plik_1.txt dopisując dodatkową linię:

Lorem ipsum dolor sit amet *consectetur adipiscing elit.*

16. Sprawdź status katalogu roboczego komendą “git status”:

```
c:\dev\git-cwiczenia>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   plik_1.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Jak widać Git wykrył, że plik_1.txt został zmodyfikowany.

17. Aby dokładniej sprawdzić co zostało zmienione w wersjonowanych plikach, możemy użyć komendy “git diff”:

```
c:\dev\git-cwiczenia>git diff
diff --git a/plik_1.txt b/plik_1.txt
index a891a0d..5b1cc64 100644
--- a/plik_1.txt
+++ b/plik_1.txt
@@ -1 +1,2 @@
-Lorem ipsum dolor sit amet
\ No newline at end of file
+Lorem ipsum dolor sit amet
+consectetur adipiscing elit.
\ No newline at end of file
```

Odpowiedź pokazuje, że różnica między stanem aktualnym a stanem z ostatniego commita to dodatkowa linia tekstu w pliku.
Systemy kontroli wersji (w tym Git) są bardzo efektywne w wersjonowaniu plików tekstowych. Gdy zapisujemy (commitujemy) kolejne wersje, nie jest tworzona kopia całego pliku (co byłoby marnowaniem przestrzeni na dysku) zapisywane są tylko niezbędne dane wynikające z różnic pomiędzy kolejnymi wersjami.

18. Wykonaj commit i sprawdź log używając poniższych komend

```
git add .
git commit -m "dodana linia tekstu"
git log
```

odpowiedzi:

```
c:\dev\git-cwiczenia>git add .

c:\dev\git-cwiczenia>git commit -m "dodana linia tekstu"
[master 4ac9377] dodana linia tekstu
 1 file changed, 2 insertions(+), 1 deletion(-)

c:\dev\git-cwiczenia>git log
commit 4ac9377de369ab6007b25c626cab381547f8ee44
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:30:08 2015 +0100

    dodana linia tekstu

commit a59e5e44f0bd2681162120d031f42b29484fa2e4
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:20:45 2015 +0100

    dodano plik_1.txt
```

Kropka w komendzie “git add” oznacza dodanie całego bieżącego katalogu czyli wszystkich zmodyfikowanych plików. W tym przypadku zmieniony mamy tylko jeden plik,
ale gdyby było ich więcej, to wszystkie dodane byłyby do staging area.
Jeżeli commit został wykonany poprawnie, to w logu zobaczymy 2 commity w kolejności od najnowszego do najstarszego.

Nazwa i email autora ostatniego commita powinny odpowiadać wcześniej skonfigurowanym wartościom.

19. Komenda “git show” pozwala sprawdzać jakie zmiany wprowadzały poszczególne commity. Aby z niej skorzystać musimy znać ID commitów, które chcemy sprawdzić. ID commita to suma kontrolna SHA1 którą zobaczymy w odpowiedzi do komendy “git log”. Przykładowo:

```
c:\dev\git-cwiczenia>git log
commit 4ac9377de369ab6007b25c626cab381547f8ee44
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:30:08 2015 +0100

    dodana linia tekstu

commit a59e5e44f0bd2681162120d031f42b29484fa2e4
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:20:45 2015 +0100

    dodano plik_1.txt
```

Zwykle nie trzeba podawać całego ID, wystarczy tylko kilka początkowych znaków (o ile jednoznacznie identyfikują one commit, tzn. nie istnieje żaden inny commit o ID zaczynającym się dokładnie taką samą sekwencją). Dodatkowo po wpisaniu kilku pierwszych znaków i naciśnięciu klawisza TAB (może nie działać w Windows) wpisywany identyfikator zostanie uzupełniony automatycznie.

```
c:\dev\git-cwiczenia>git show a59e5e44f0bd2681162120d031f42b29484fa2e4
commit a59e5e44f0bd2681162120d031f42b29484fa2e4
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:20:45 2015 +0100

    dodano plik_1.txt

diff --git a/plik_1.txt b/plik_1.txt
new file mode 100644
index 0000000..a891a0d
--- /dev/null
+++ b/plik_1.txt
@@ -0,0 +1 @@
+Lorem ipsum dolor sit amet
\ No newline at end of file
```

```
c:\dev\git-cwiczenia>git show 4ac9377de369ab6007b25c626cab381547f8ee44
commit 4ac9377de369ab6007b25c626cab381547f8ee44
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:30:08 2015 +0100

    dodana linia tekstu

diff --git a/plik_1.txt b/plik_1.txt
index a891a0d..5b1cc64 100644
--- a/plik_1.txt
+++ b/plik_1.txt
@@ -1 +1,2 @@
-Lorem ipsum dolor sit amet
\ No newline at end of file
+Lorem ipsum dolor sit amet
+consectetur adipiscing elit.
\ No newline at end of file
```

20. Aby przywrócić stan katalogu roboczego do wersji zapisanej w dowolnym commicie używamy komendy “git checkout”. Wymaga ona podania ID commita (lub początkowy fragment ID, obowiązują te same zasady co dla poprzednio opisanej komendy). Przykładowo:

```
c:\dev\git-cwiczenia>git checkout a59e5e44f0bd2681162120d031f42b29484fa2e4
Note: checking out 'a59e5e44f0bd2681162120d031f42b29484fa2e4'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at a59e5e4... dodano plik_1.txt
```

Zbadaj stan katalogu roboczego. Zawartość pliku plik_1.txt powinna odpowiadać pierwotnej wersji, tzn z jedną linią tekstu.

21. Wróć do najnowszej wersji używając komendy “git checkout” z ID ostatniego commita lub podając nazwę brancha (master):

```
c:\dev\git-cwiczenia>git checkout master
Previous HEAD position was a59e5e4... dodano plik_1.txt
Switched to branch 'master'
```

Branch i merge

1. Utwórz nowe repozytorium korzystając z komendy “git init” tak jak w poprzednim ćwiczeniu.

2. Utwórz plik tekstowy plik.txt z poniższą zawartością:

aaa *bbb* *ccc*

3. Wykonaj commit powyższego pliku (git add, git commit).

```
c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "dodano plik.txt"
[master (root-commit) d366080] dodano plik.txt
 1 file changed, 3 insertions(+)
 create mode 100644 plik.txt
```

4. Utwórz nowy branch o nazwie “testowy”:

```
c:\dev\git-cwiczenia2>git branch testowy
```

5. Przejdź na nowo utworzony branch:

```
c:\dev\git-cwiczenia2>git checkout testowy
Switched to branch 'testowy'
```

6. Dopisz kolejną linię (“ddd”) do pliku tekstowego. Wykonaj commit.

```
c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "zmieniono plik.txt"
[testowy 1417a70] zmieniono plik.txt
 1 file changed, 2 insertions(+), 1 deletion(-)
```

7. Przejdź z powrotem na branch master. Sprawdź zawartość pliku ostatnia zmiana nie jest widoczna, ponieważ była wykonana na innym branchu. Podobnie komenda “git log” nie pokaże commitów wykonanych na innych branchach niż aktualny (czyli

master).

```
c:\dev\git-cwiczenia2>git checkout master
Switched to branch 'master'
```

```
c:\dev\git-cwiczenia2>git log
commit 152d9e046263c2dba587d561583ae43d7e82a1ae
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:58:14 2015 +0100

    dodano plik.txt
```

8. Wykonaj merge brancha “testowy”:

```
c:\dev\git-cwiczenia2>git merge testowy
Updating 152d9e0..1417a70
Fast-forward
 plik.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Używając komendy “git merge” podajemy nazwę brancha który chcemy połączyć z aktualnie aktywnym branchem (czyli tym, który wyświetli się po użyciu “git status”).

9. Branch “testowy” był o 1 commit dalej niż branch “master”. Na branchu “master” nie było żadnych innych zmian od punktu rozgałęzienia, więc efektem merga jest po prostu dodanie ostatniego commita do brancha “master”. Zweryfikuj to komendą “git log” wynik powinien być identyczny na obu branchach.

Branch master:

```
c:\dev\git-cwiczenia2>git log
commit 1417a7013f40f37b593b10b0c5f57db253d669b9
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:59:12 2015 +0100

    zmieniono plik.txt

commit 152d9e046263c2dba587d561583ae43d7e82a1ae
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:58:14 2015 +0100

    dodano plik.txt
```

Branch testowy:

```
c:\dev\git-cwiczenia2>git checkout testowy
Switched to branch 'testowy'

c:\dev\git-cwiczenia2>git log
commit 1417a7013f40f37b593b10b0c5f57db253d669b9
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:59:12 2015 +0100

    zmieniono plik.txt

commit 152d9e046263c2dba587d561583ae43d7e82a1ae
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:58:14 2015 +0100

    dodano plik.txt
```

10. Ponownie przełącz się na branch “testowy” i dodaj kolejną linię (“eee”). Wykonaj commit.

```
c:\dev\git-cwiczenia2>git checkout testowy
Already on 'testowy'

c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "eee plik.txt"
[testowy 69c98ed] eee plik.txt
 1 file changed, 2 insertions(+), 1 deletion(-)
```

11. Wróć do brancha “master”. Utwórz nowy plik tekstowy z dowolną zawartością. Wykonaj commit.

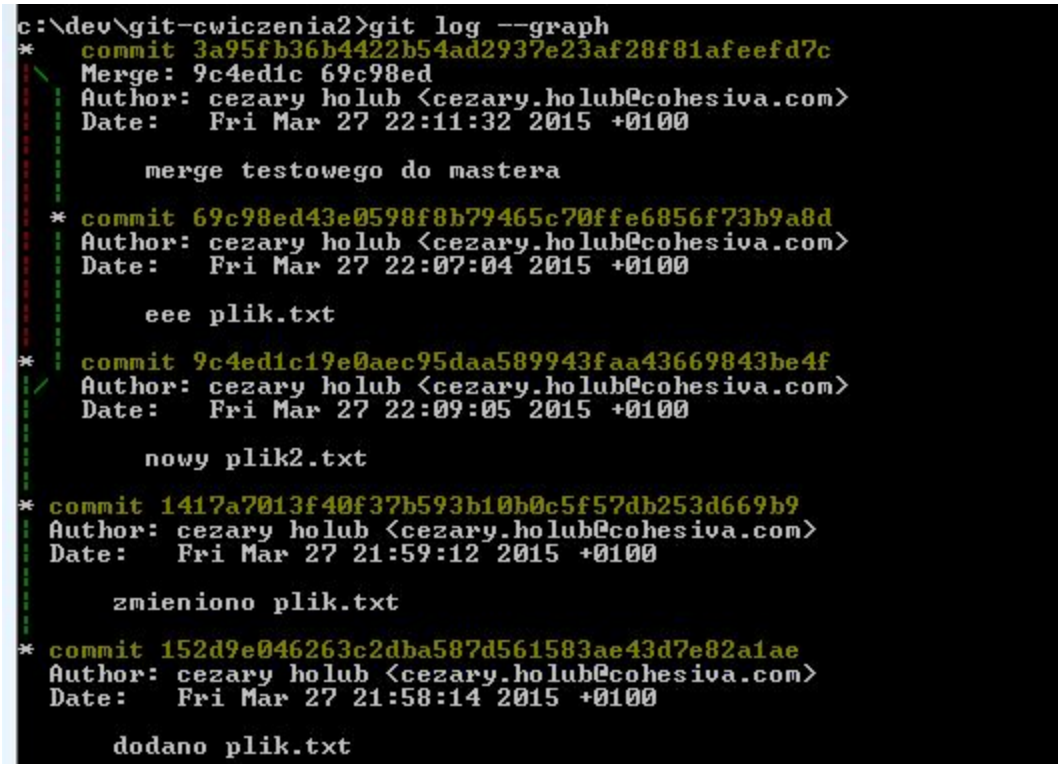
```
c:\dev\git-cwiczenia2>git checkout master
Switched to branch 'master'

c:\dev\git-cwiczenia2>git add plik2.txt

c:\dev\git-cwiczenia2>git commit -m "nowy plik2.txt"
[master 9c4ed1c] nowy plik2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 plik2.txt
```

12. W tym momencie od ostatniego mergea zarówno na branchu “master” jak i “testowy” wykonany został commit. Z tego powodu w wyniku “git merge” dodany zostanie dodatkowy commit łączący oba branche. Jego opis można podać używając flagi m jak poniżej:

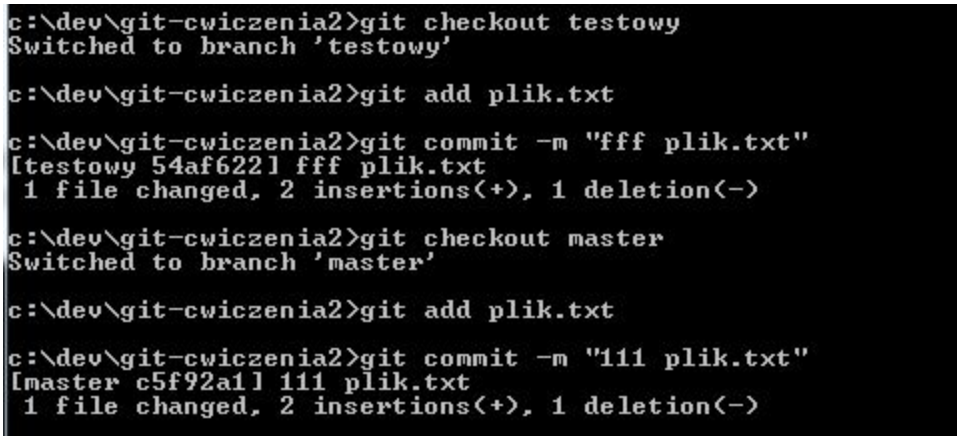
```
c:\dev\git-cwiczenia2>git merge testowy -m "merge testowego do mastera"
Merge made by the 'recursive' strategy.
 plik.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

13. Zbadaj wynik operacji za pomocą “git log --graph”. Branch master posiada teraz zmiany z brancha testowy, ale połączenie branchy wymagało dodatkowego commita.

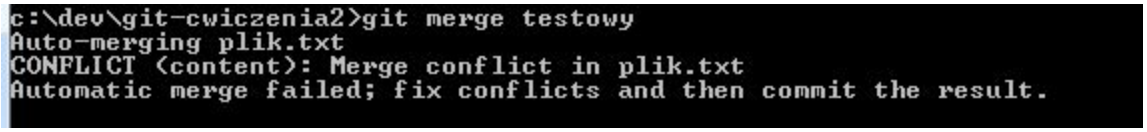
Dodanie flagi graph do komendy “git log” lepiej obrazuje sytuację:



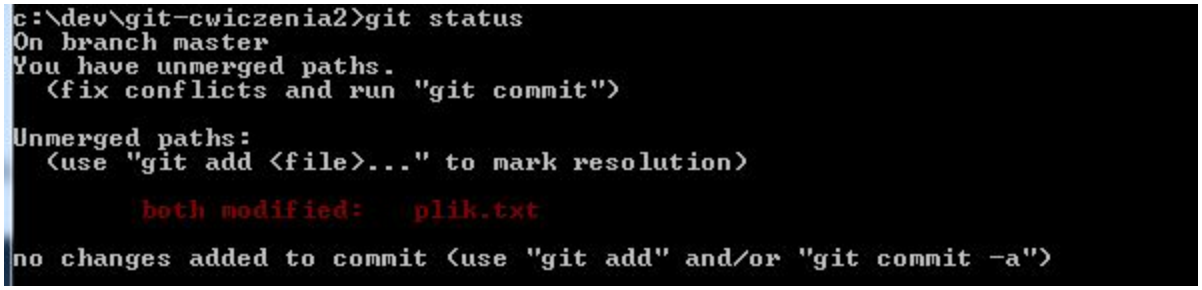
14. W poprzednim przypadku commit mergeujący mógł być wykonany automatycznie ponieważ zmiany na obu branchach dotyczyły różnych plików. Zmiany w tym samym pliku często również mogą być łączone automatycznie o ile nie są wykonane w tej samej linii. W takim przypadku mamy konflikt, który trzeba rozwiązać ręcznie. Aby zasymulować konfliktową sytuację, przełącz się na branch “testowy”. Dopisz do pierwszego pliku kolejną linię “fff” i wykonaj commit. Wróć na branch “master” i tym samym pliku dopisz linię “111”. Wykonaj commit.



15. Spróbuj wykonać merge brancha “testowy”. Pojawi się informacja o konflikcie:



16. Git jest w trybie rozwiązywania konfliktu. Wykonaj “git status” by poznać szczegóły:



Odpowiedź programu informuje nas w którym plik jest konflikt. W tym przypadku nie jest to zaskoczeniem, ale w rzeczywistym projekcie konflikty wynikają najczęściej gdy łączone są zmiany wykonane przez 2 różne osoby nie zawsze jest łatwe do przewidzenia gdzie i kiedy wystąpi konflikt.

17. Git oczekuje od użytkownika rozwiązania konfliktu (poprzez ręczną edycję pliku) po czym wykonania commita (będzie to merge commit tak jak poprzednio, z tą różnicą że nie może on być teraz wykonany automatycznie). Aby rozwiązać konflikt otwórz (np. w Notatniku) plik w którym on wystąpił. Jego zawartość będzie podobna do poniższej:

```
aaa
bbb
ccc
ddd
eee
<<<<<<< HEAD
111
=======
fff
>>>>>>> testowy
```

Znaczniki tekstowe pokazują obydwie wersje odpowiednio “111” z mastera (oznaczone jako HEAD, bo branch master jest aktualnie aktywny) oraz “fff” z testowego. Zadaniem użytkownika jest sprowadzenie pliku do żądanej postaci Czasami

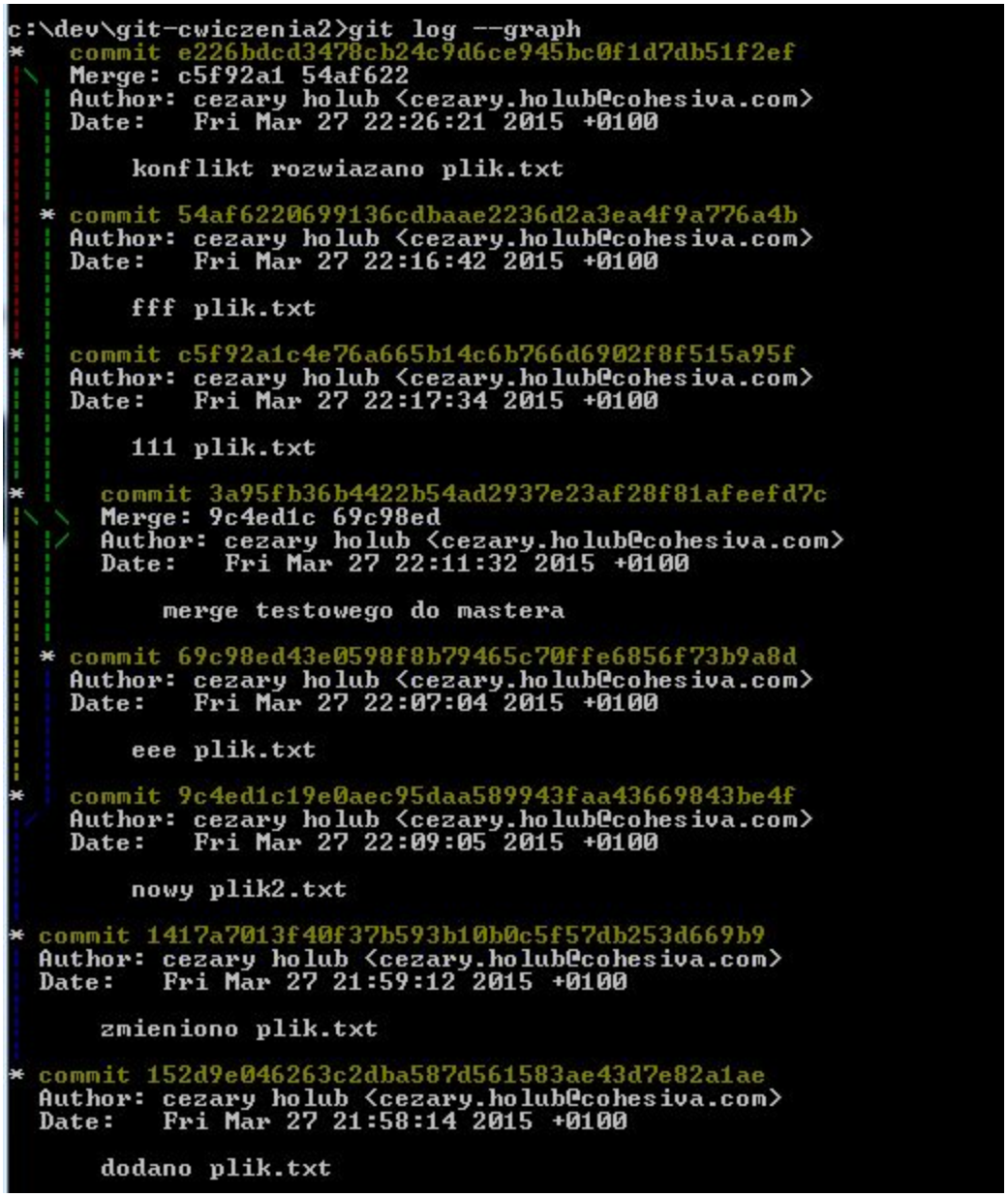
chcemy po prostu wybrać jedną z wersji usuwamy wtedy pozostałe linie. Czasami obydwie zmiany wnoszą coś istotnego wtedy musimy sami je połączyć w żądany tekst. Na potrzeby tego ćwiczenia załóżmy że obydwie zmiany są dla nas istotne, przy czym "fff" ma być przed "111". Plik powinien mieć zatem taką zawartość:

aaa
bbb
ccc
ddd
eee
fff
111

18. Po zapisaniu pliku wykonaj commit standardowymi komendami (git add . oraz git commit -m "opis").

```
c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "konflikt rozwiazano plik.txt"
[master e226bdc] konflikt rozwiazano plik.txt
```

19. Zbadaj efekt komendą "git log graph"



Praca z repozytorium zdalnym (na ocenę)

Następne laboratorium