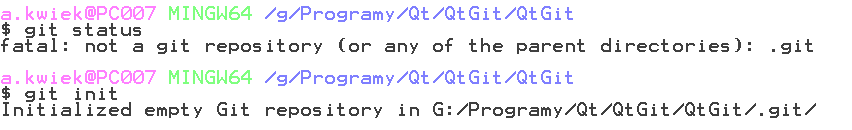
# INICJOWANIE REPOZYTORIUM

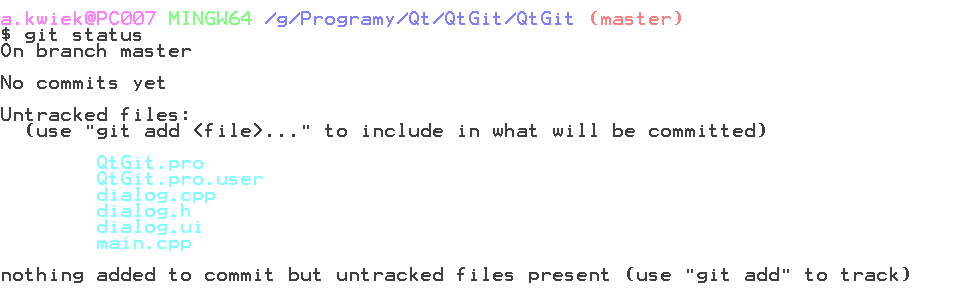
⇒ Utworzyć i zbudować nowy projekt Qt - QDialog

* Zainicjowanie repozytorium w katalogu roboczym (pliki źródłowe projektu)

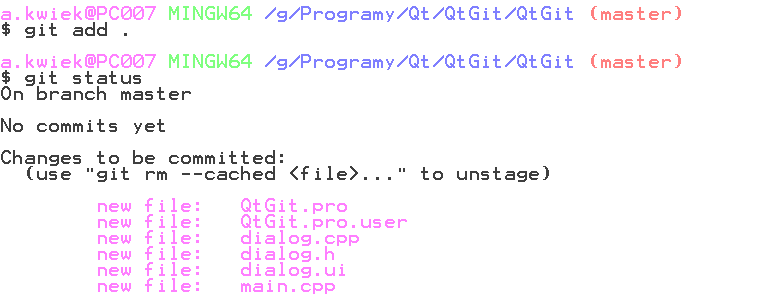


# PUNKTY KONTROLI WERSJI OPROGRAMOWANIA (commit)

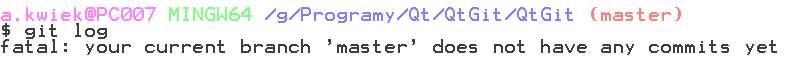
* Sprawdzenie status plików katalogu roboczego



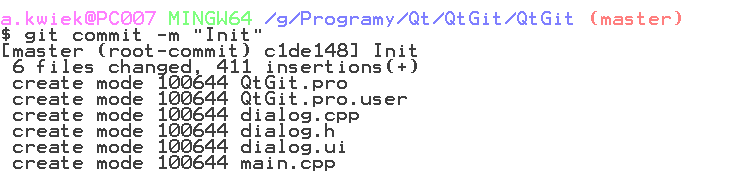
* Przygotowanie commitu (dodanie plików do stage) i ponowne sprawdzenie statusu



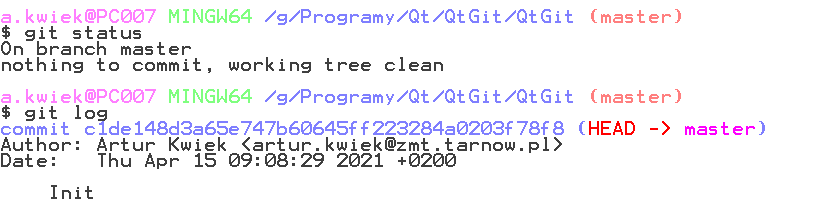
* Sprawdzenie historii commitów (*brak*)



* Wykonanie commit-u



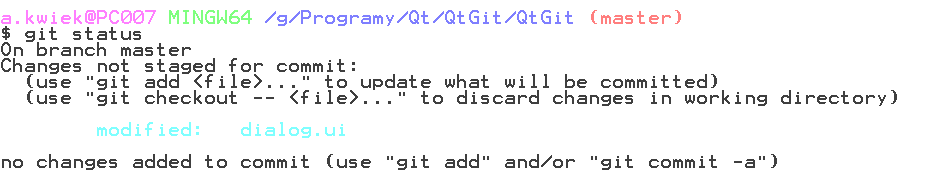
* Sprawdzenie statusu i historii commitów



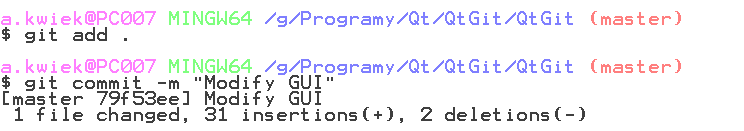
# ZAPISYWANIE ZMIAN (commit)

⇒ Dodać elementy i zmodyfikować interfejs - QButton, QLineEdit, QLabel (status, log, zawartość katalogu)

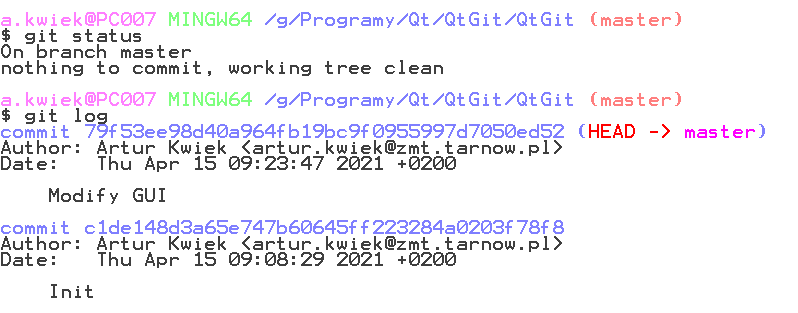
* Sprawdzenie statusu plików źródłowych projektu (status plików projektu)



* Przygotowanie plików do commitu i zapisanie punktu kontrolnego stanu projektu (commit)



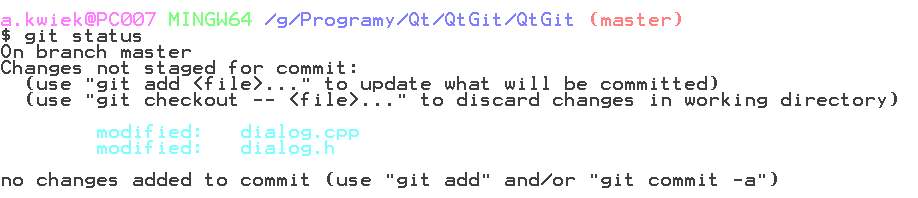
* Sprawdzenie statusu i historii commitów (status, log, zawartość katalogu)



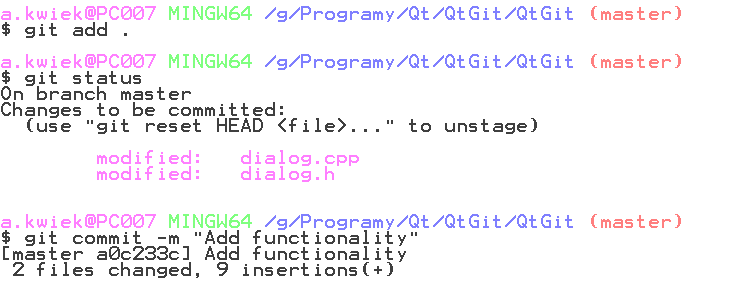
⇒ Dodać funkcjonalność

→ Po wpisaniu tekstu w textEdit i naciśnięciu buttona, tekst ląduje na etykiecie. (status, log, zawartość katalogu)

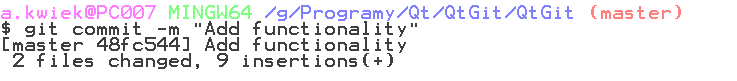
* Sprawdzenie statusu plików projektu



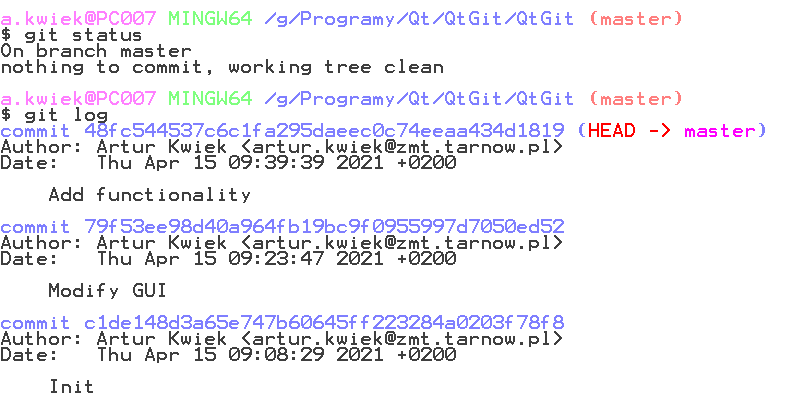
* Przygotowanie plików do commitu (dodanie na stage)



* Commit



* Sprawdzenie statusu i przegląd commitów



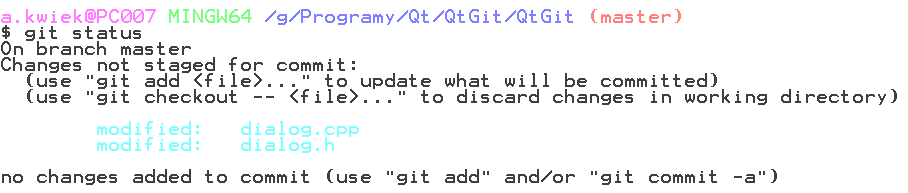
# COFANIE ZMIAN

<https://cezarywalenciuk.pl/blog/programing/triki-z-git--reset-restore-revert-czyli-cofanie>

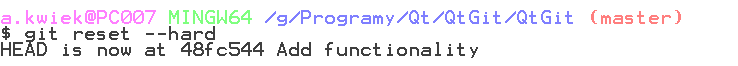
⇒ Wprowadzić zmiany w projekcie

→ Po wpisaniu tekstu w pole edit tekst kopiowany jest automatycznie do etykiety label

→ Po naciśnięciu buttona pole tekstowe edit jest czyszczone

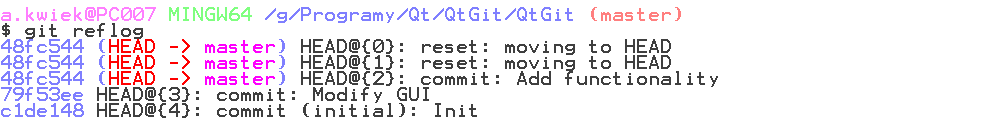


* Cofnięcie bieżących zmian w plikach



*Teraz zawartość plików jest taka jak przed zmianami (po commicie „Add functionality”)*

* Cofnięcie zmian do wcześniejszego commitu
* Wypisanie skróconych tagów

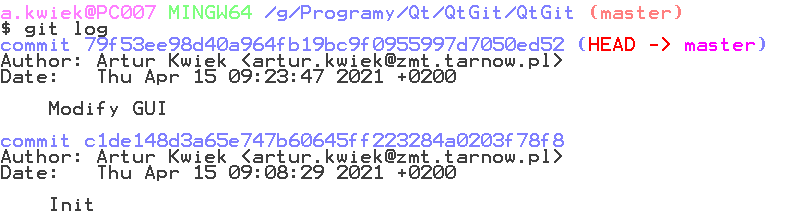


*Poruszanie się po historii możliwe jest przez polecenie checkout lub git reset --soft*

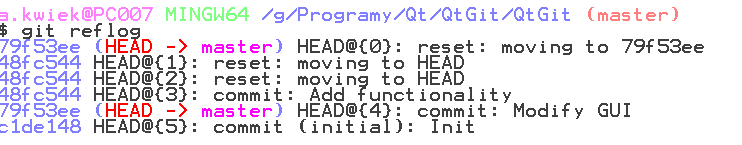
* Powrót do wcześniejszej wersji (stan projektu bez funkcjonalności) (reflog, reset, status, log, zawartość katalogu)



*Bieżąca historia commitów odwołuje się do aktualnego dowiązania (log)*



*Ale pełna historia zmian w repozytorium ciągle jest dostępna*

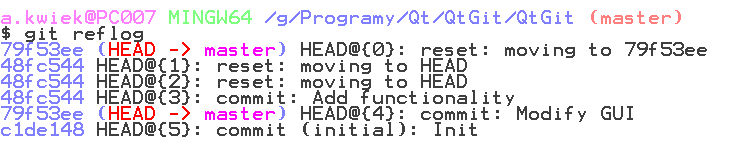


# GAŁĘZIE

* Przegląd gałęzi repozytorium



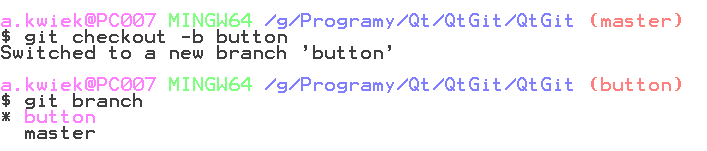
* Powrót do wcześniejszej wersji (Initial) (status, log, zawartość katalogu)



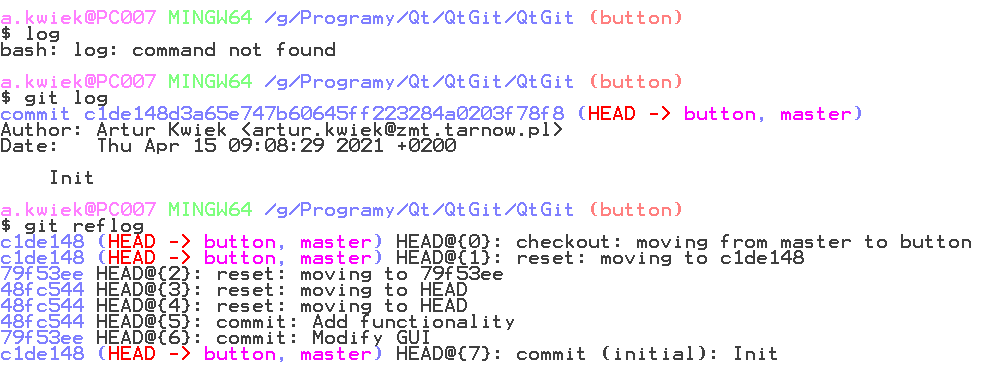


*W tym stanie mamy pusty dialog, bez dodanych wcześniej widgetów*

* Utworzenie gałęzi button i przełączenie się na nią (checkout -b, branch, status, log)

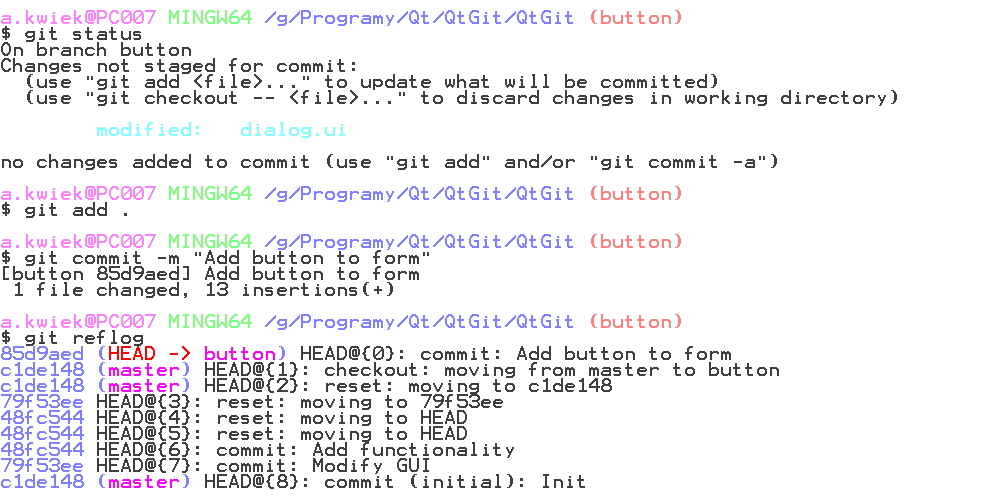


*Nowa gałąź po utworzeniu posiada taką samą historię jak gałąź główna*

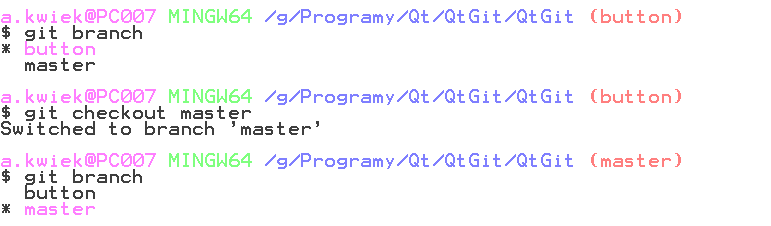


⇒ Dodać do formatki przycisk i wykonać commit

*Na gałęzi tworzona jest dalsza historia zmian plików*

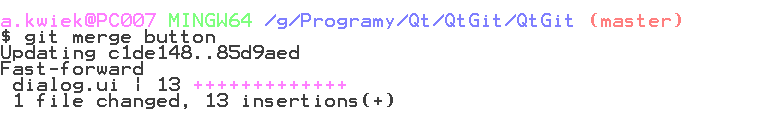


* Przełączenie się na gałąź master.



*Teraz ponownie mamy stan z pustą formatką*

* Połączenie zmian na gałęzi button z aktualnym stanem gałęzi master



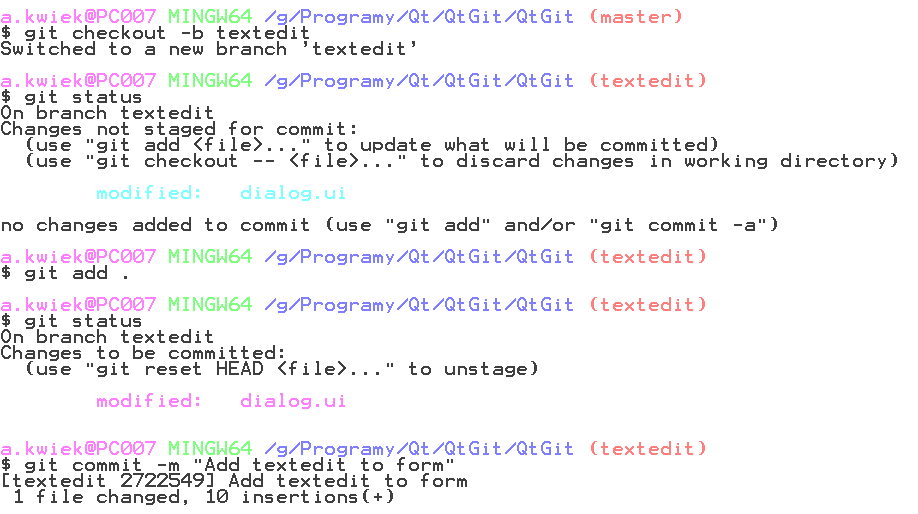
*Zmiany z gałęzi button zostały naniesione na gałąź główną*

# KONFLIKTY

* Utworzenie gałęzi textedit i przełączenie się na nią (checkout -b, branch, status, log)

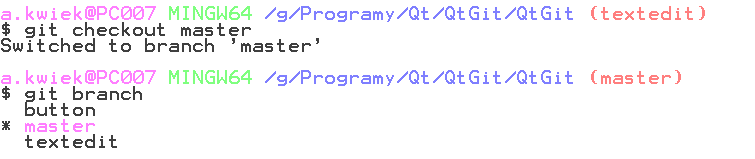


⇒ Dodać do formatki pole tekstowe QTextEdit i wykonać commit

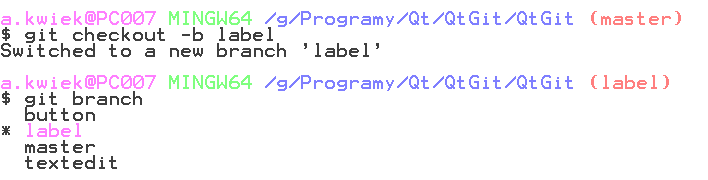


* Przełączenie się na gałąź master.

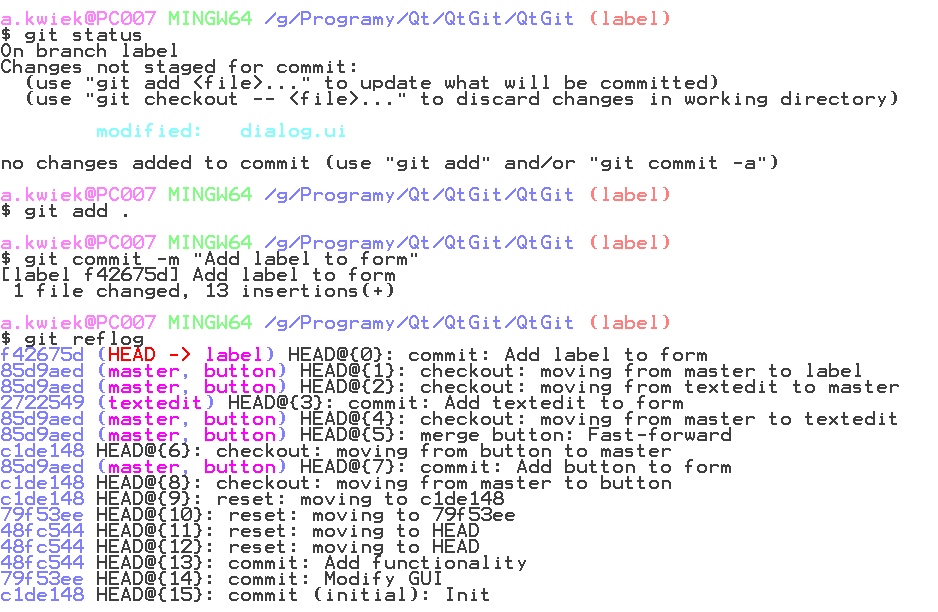
*Teraz w repozytorium występują trzy gałęzie, a jego stan bieżący dowiązany jest do gałęzi master*



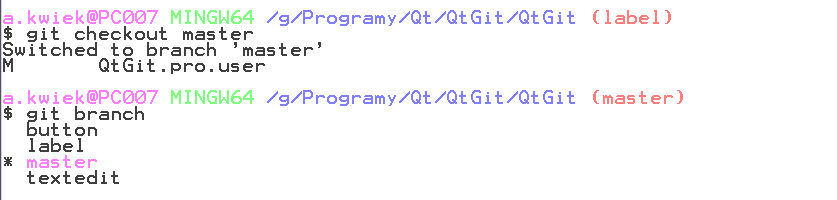
* Utworzenie gałęzi label i przełączenie się na nią (checkout -b, branch, status, log)



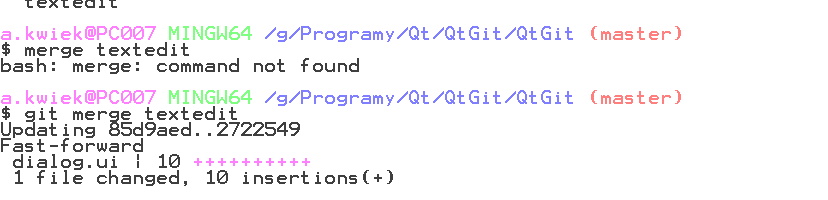
⇒ Dodać do formatki pole tekstowe QLabel i wykonać commit (wyświetlić bieżącą historię zmian w repozytorium)



* Przełączenie się na gałąź master i wyświetlenie gałęzi repozytorium projektu.

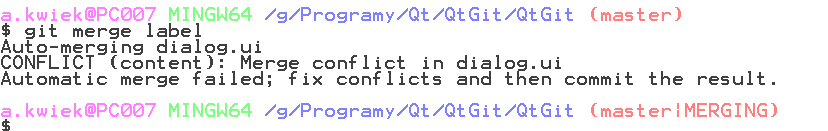


* Połączenie zmian na gałęzi textedit z gałęzią główną



*Na formatce projektu pojawiło się pole textowe textedit*

* Próba połączenia zmian na gałęzi label z gałęzią główną



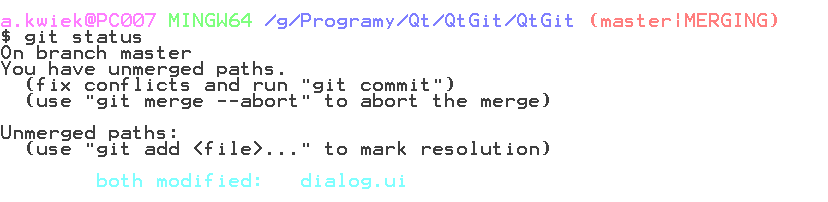
*Położenie w repozytorium zmieniło się na nową gałąź master | MERGING*

*System git zgłasza wystąpienie konfliktu. Wynika on z faktu iż nowe zmiany wprowadzone będą w miejscu, które zawiera inne modyfikacje pochodzące z połączenia z gałęzią textedit.*

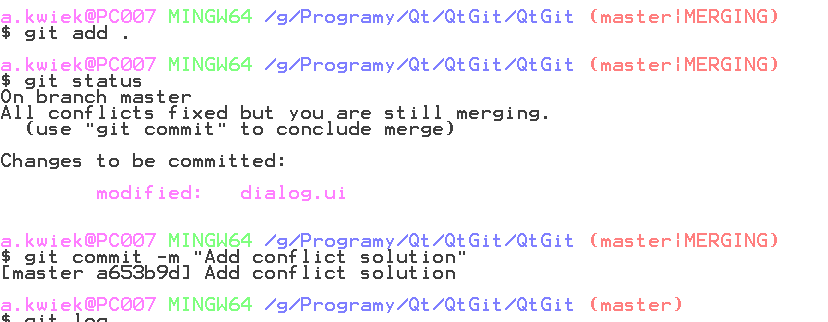
⇒ Rozwiązać konflikty

*Po rozwiązaniu konfliktów sprawdzamy status plików projektu*

* Sprawdzenie statusu plików projektu

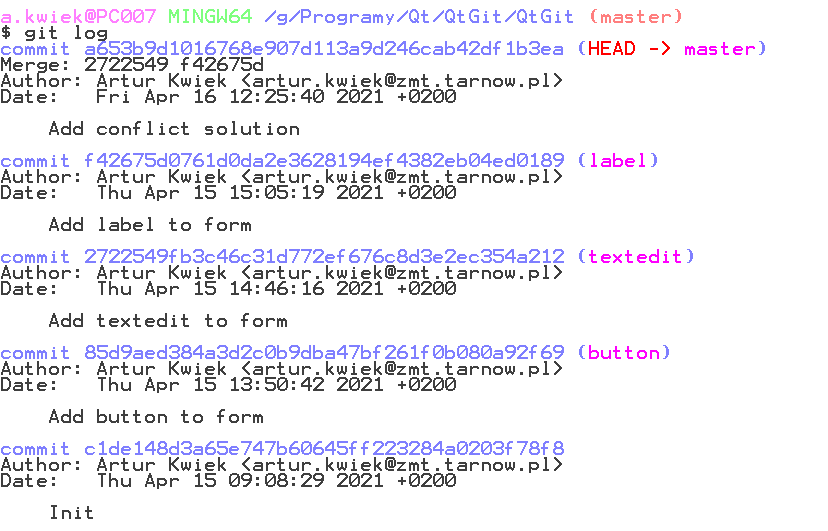


* Przygotowujemy pliki i wykonujemy commit



*Projekt przełącza się na gałąź master*

* Sprawdzenie historii comitów



*Widzimy, że zawiera wszystkie zmiany*

* Sprawdzenie historii repozytorium



*Pokazuje pełną historię zmian*

* Dodanie funkcjonalności:

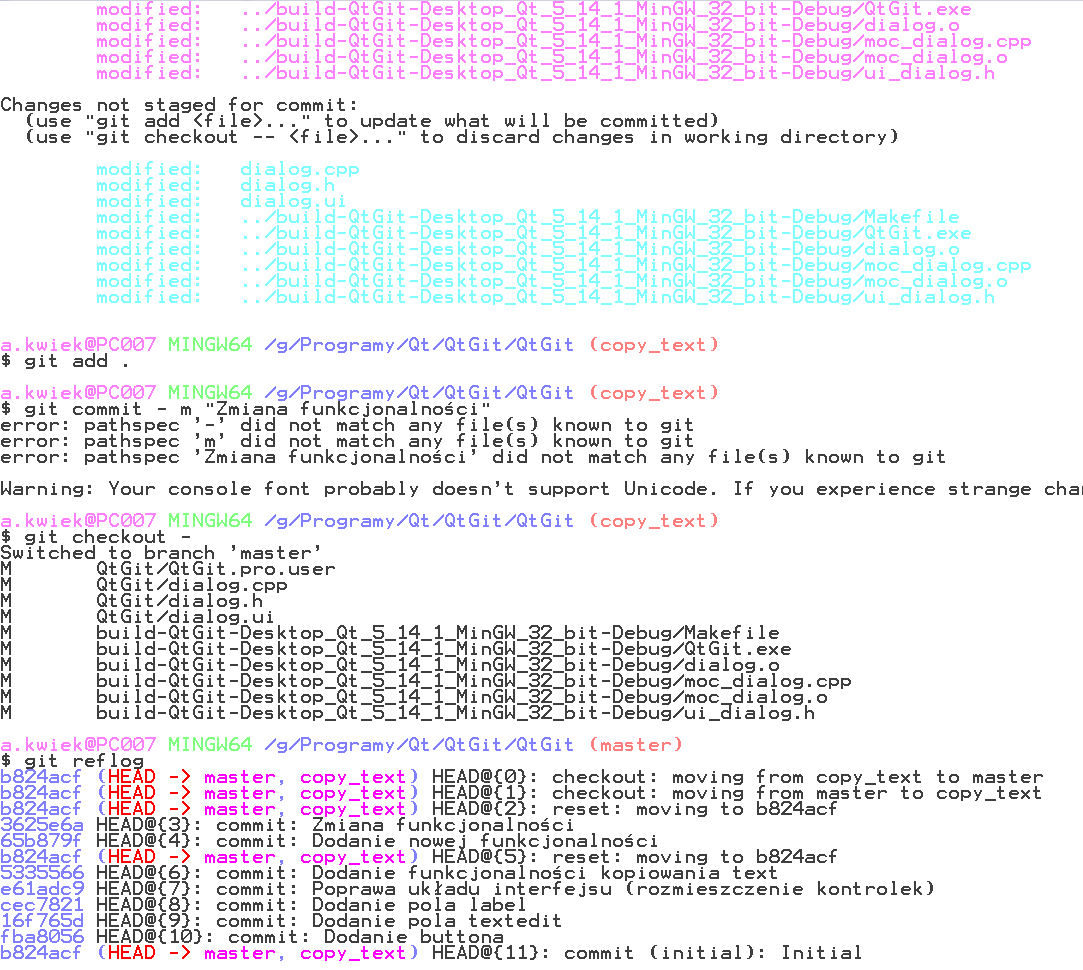
⇒ Wprowadzić zmiany w projekcie

→ Po wpisaniu naciśnięciu buttona, text w label powiększany jest o znak # (status, log, zawartość katalogu)

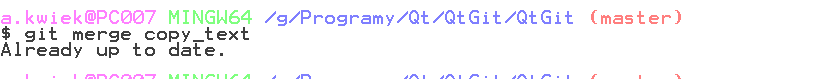
* Commit (status, log, zawartość katalogu)



* Przełączenie się na master (status, log, zawartość katalogu, sprawdzenie galezi)



* Połączenie funkcjonalności(status, log, zawartość katalogu)



* Powrót do wcześniejszej wersji (bez funkcjonalności)
* Utworzenie gałęzi textcopy i przełączenie się na nią
* git status (czy potrzebny commit?)
* Dodanie funkcjonalności:

- Po wpisaniu textu w textEdit text ląduje na etykiecie. (status, log, zawartość katalogu)

* Przełączenie na master. (status, log, zawartość katalogu)
* Utworzenie gałęzi textclear i przełączenie się na nią(status, log, zawartość katalogu)
* git status (czy potrzebny commit?)
* Dodanie funkcjonalności:

- Po naciśnięciu buttona text w textEdit jest czyszczony. (status, log, zawartość katalogu)

* Przełączenie na master.
* Połączenie funkcjonalności z gałęzi textcopy. (status, log, zawartość katalogu)
* Próba połączenia funkcjonalności z textclear. (status, log, zawartość katalogu)

----------------