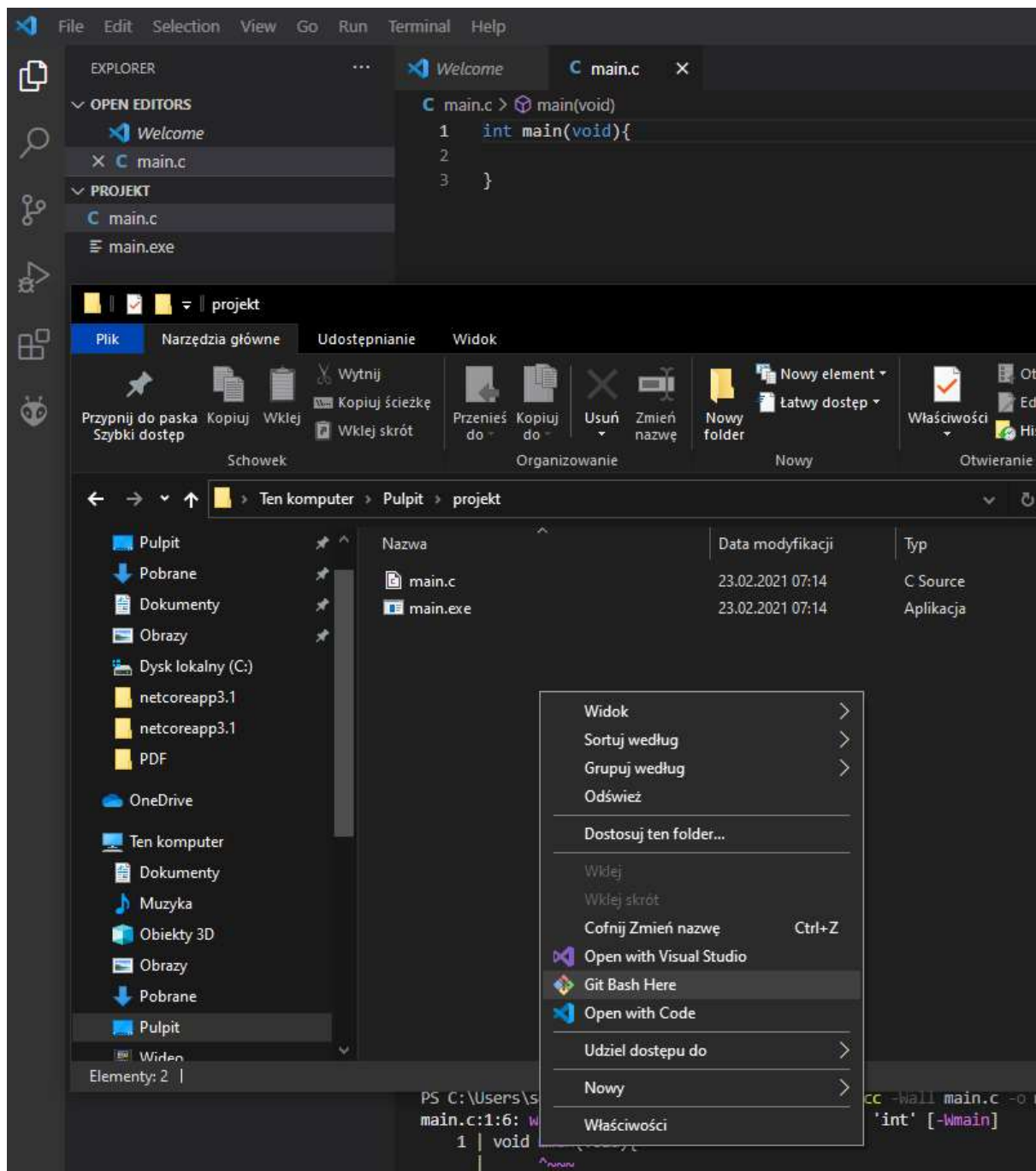


Zawartość:

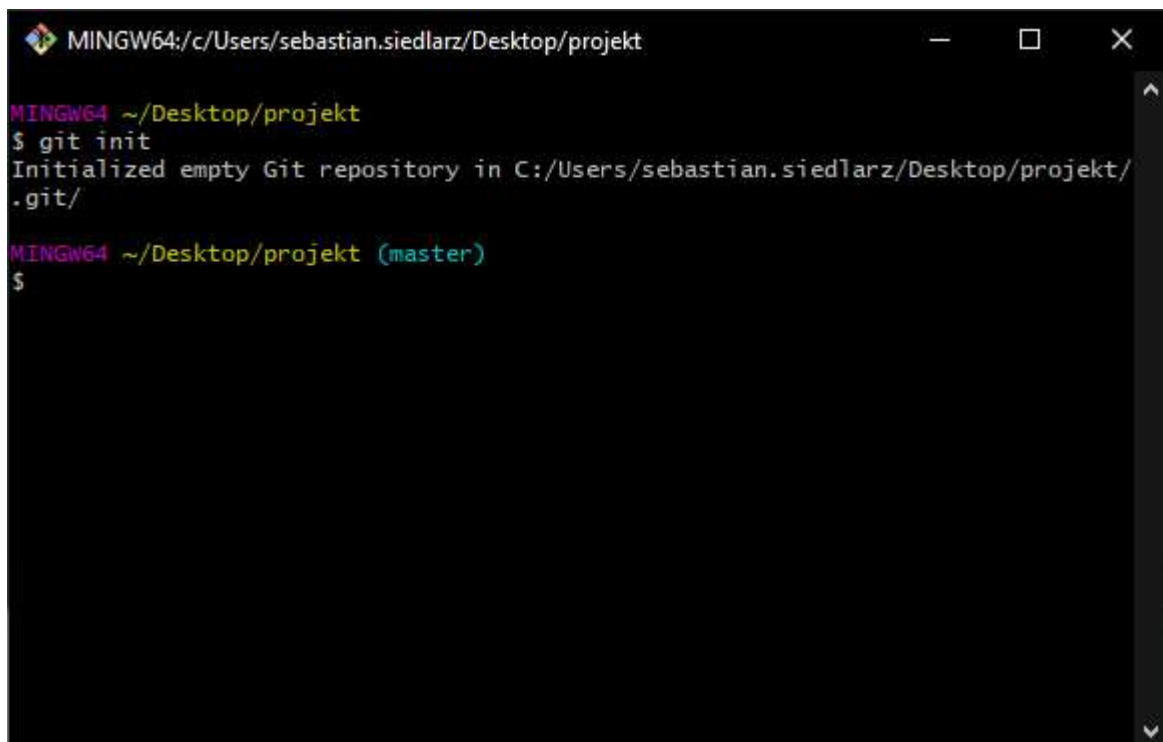
- inicjalizacja repozytorium lokalnego
- commitowanie zmian
- przeglądanie historii zmian
- schowek
- cofanie zmian
- gałęzie
- zdalne repozytorium

Inicjalizacja repozytorium



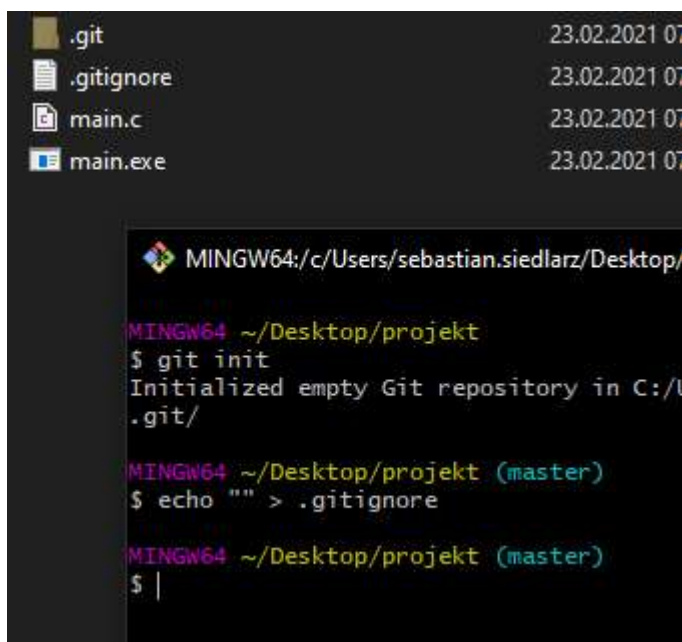
Inicjujemy repozytorium lokalne za pomocą polecenia:

git init



```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
$ git init
Initialized empty Git repository in C:/Users/sebastian.siedlarz/Desktop/projekt/.git/
MINGW64 ~/Desktop/projekt (master)
$
```

Warto stworzyć plik .gitignore po to by git nie zbierał niechcianych plików jak np. pliki binarne



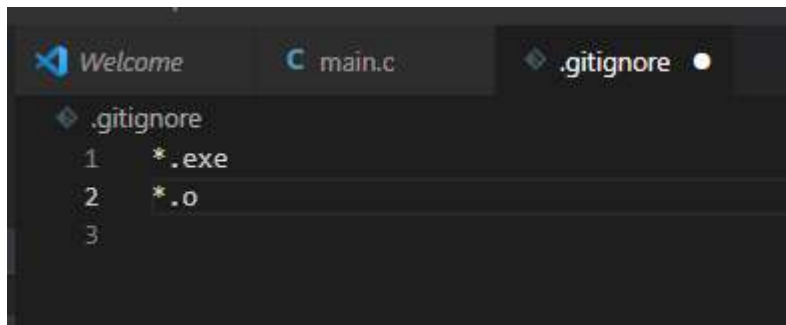
```

.git
.gitignore
main.c
main.exe

MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
$ git init
Initialized empty Git repository in C:/Users/sebastian.siedlarz/Desktop/projekt/.git/
MINGW64 ~/Desktop/projekt (master)
$ echo "" > .gitignore
MINGW64 ~/Desktop/projekt (master)
$ |
```

W pliku tym wskazujemy pliki oraz foldery do pominięcia. W sieci znaleźć można gotowe pliki gitignore dla środowisk takich jak Visual Studio czy Android Studio

<https://github.com/github/gitignore>



Dodawanie plików do śledzenia

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt

MINGW64 ~/Desktop/projekt
$ git init
Initialized empty Git repository in C:/Users/sebastian.siedlarz/Desktop/projekt/.git/

MINGW64 ~/Desktop/projekt (master)
$ echo "" > .gitignore

MINGW64 ~/Desktop/projekt (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory

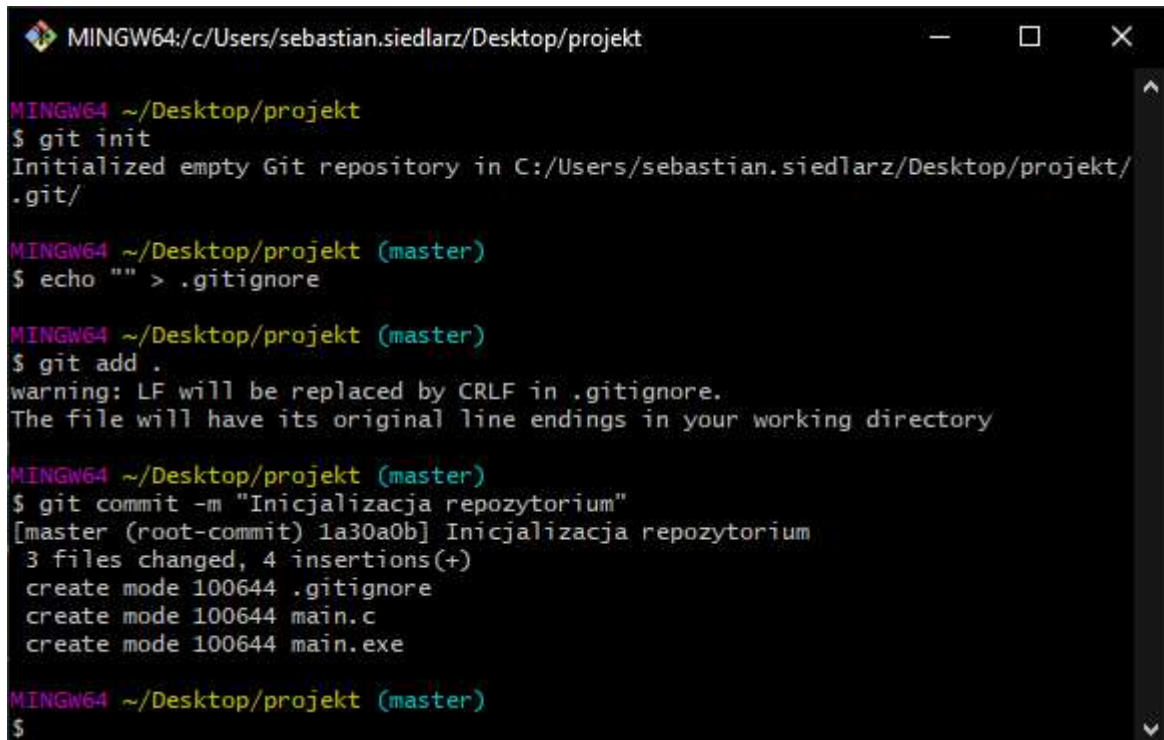
MINGW64 ~/Desktop/projekt (master)
$
```

Dodanie wszystkich plików poleceniem:
git add .

lub pojedyncze

git add nazwa

Cofanie zmian w repozytorium

A screenshot of a Windows command prompt window titled "MINGW64: c:/Users/sebastian.siedlarz/Desktop/projekt". The window shows the following commands and output:

```
MINGW64 ~/Desktop/projekt
$ git init
Initialized empty Git repository in C:/Users/sebastian.siedlarz/Desktop/projekt/.git/

MINGW64 ~/Desktop/projekt (master)
$ echo "" > .gitignore

MINGW64 ~/Desktop/projekt (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory

MINGW64 ~/Desktop/projekt (master)
$ git commit -m "Inicjalizacja repozytorium"
[master (root-commit) 1a30a0b] Inicjalizacja repozytorium
3 files changed, 4 insertions(+)
create mode 100644 .gitignore
create mode 100644 main.c
create mode 100644 main.exe

MINGW64 ~/Desktop/projekt (master)
$
```

Polecenie:

git commit -m "Komentarz"

W przypadku braku przełącznika -m otworzy się domyślny edytor tekstu ustawiony w trakcie instalacji. Wystarczy wprowadzić treść, zapisać i zamknąć edytor.

Historia commitów

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
MINGW64 ~/Desktop/projekt (master)
$ git add .

MINGW64 ~/Desktop/projekt (master)
$ git commit -m "Helloworld"
[master 4d38c66] Helloworld
1 file changed, 3 insertions(+), 1 deletion(-)

MINGW64 ~/Desktop/projekt (master)
$ git log
commit 4d38c661a45cbcd4ecf1286cd149b8c9ae809283 (HEAD -> master)
Author: Sebastian Siedlarz <sebastian.siedlarz@vitberg.com>
Date: Tue Feb 23 07:59:31 2021 +0100

    Helloworld

commit 1a30a0b1c3089f7224f49ac48061dc0aa54bfcd8
Author: Sebastian Siedlarz <sebastian.siedlarz@vitberg.com>
Date: Tue Feb 23 07:47:31 2021 +0100

    Inicjalizacja repozytorium

MINGW64 ~/Desktop/projekt (master)
$
```

Polecenie do wyświetlania historii commitów:

git log

lub bardziej przystępna forma:

git log --oneline

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
[master 4d38c66] Helloworld
1 file changed, 3 insertions(+), 1 deletion(-)

MINGW64 ~/Desktop/projekt (master)
$ git log
commit 4d38c661a45cbcd4ecf1286cd149b8c9ae809283 (HEAD -> master)
Author: Sebastian Siedlarz <sebastian.siedlarz@vitberg.com>
Date: Tue Feb 23 07:59:31 2021 +0100

    Helloworld

commit 1a30a0b1c3089f7224f49ac48061dc0aa54bfcd8
Author: Sebastian Siedlarz <sebastian.siedlarz@vitberg.com>
Date: Tue Feb 23 07:47:31 2021 +0100

    Inicjalizacja repozytorium

MINGW64 ~/Desktop/projekt (master)
$ git log --oneline
4d38c66 (HEAD -> master) Helloworld
1a30a0b Inicjalizacja repozytorium

MINGW64 ~/Desktop/projekt (master)
$
```


Polecenie:

git status

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
no changes added to commit (use "git add" and/or "git commit -a")

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.c

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

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.c

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

MINGW64 ~/Desktop/projekt (master)
$ |
```

Na czerwono zaznaczono pliki nie dodane na "stage"

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.c

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

MINGW64 ~/Desktop/projekt (master)
$ git add .

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.c

MINGW64 ~/Desktop/projekt (master)
$ |
```

Po dodaniu pliku na "stage" poleceniem **git add .** lub **git add main.c** plik zostaje oznaczony jako zielony i można na nim wykonać commit

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt

MINGW64 ~/Desktop/projekt (master)
$ git add .

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.c

MINGW64 ~/Desktop/projekt (master)
$ git commit -m "Dodanie zmiennej globalnej"
[master 4f12169] Dodanie zmiennej globalnej
1 file changed, 2 insertions(+)

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 ~/Desktop/projekt (master)
$
```

Po wykonaniu commit'a **git status** nie wskazuje, żadnych plików

Wysyłanie kodu do zdalnego repo

Dodajemy link do repozytorium zdalnego:

git remote add origin <https://github.com/user/repo.git>

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.c

MINGW64 ~/Desktop/projekt (master)
$ git commit -m "Dodanie zmiennej globalnej"
[master 4f12169] Dodanie zmiennej globalnej
1 file changed, 2 insertions(+)

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

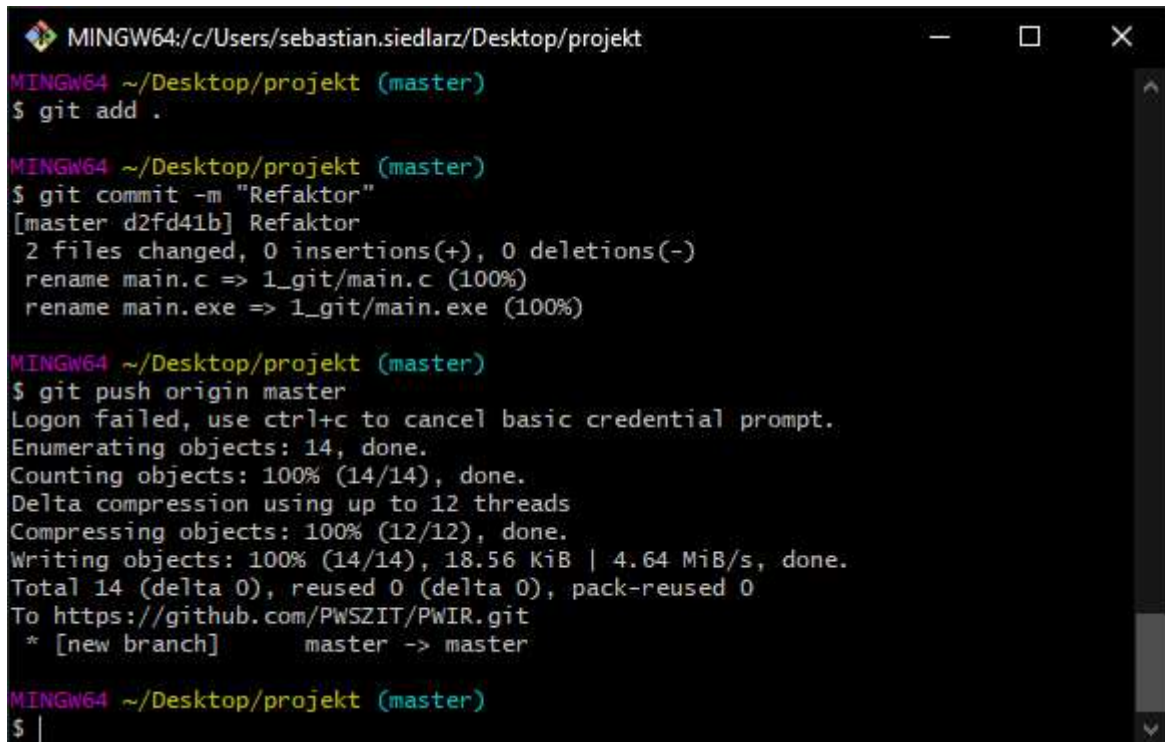
MINGW64 ~/Desktop/projekt (master)
$ git remote add origin https://github.com/PWSZIT/PWIR.git

MINGW64 ~/Desktop/projekt (master)
$
```


Dodany odnośnik możemy usunąć poleceniem by np. dodać inny:
git remote remove origin

Wypychanie zmian wykonuje się poleceniem:
git push origin nazwa_gałęzi -> git push origin master

lub wypychanie wszystkich gałęzi:
git push --all origin



```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
MINGW64 ~/Desktop/projekt (master)
$ git add .

MINGW64 ~/Desktop/projekt (master)
$ git commit -m "Refaktor"
[master d2fd41b] Refaktor
2 files changed, 0 insertions(+), 0 deletions(-)
rename main.c => 1_git/main.c (100%)
rename main.exe => 1_git/main.exe (100%)

MINGW64 ~/Desktop/projekt (master)
$ git push origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 18.56 KiB | 4.64 MiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/PWSZIT/PWIR.git
 * [new branch]      master -> master

MINGW64 ~/Desktop/projekt (master)
$ |
```

Gałęzie

Po co:

- gdy zachodzi potrzeba prowadzenia kilku wersji projektu
- gdy potrzebujemy przetestować jakieś eksperymentalne zmiany
- dobrym zwyczajem jest trzymanie na głównej gałęzi stabilnej wersji projektu a rozwijanie go na innej (gdy skończymy pracę i przetestujemy kod robimy "merge" i kontynuujemy pracę w analogiczny sposób)

Tworzenie brancha na bazie istniejącego:

Przykład tworzymy branch o nazwie "Development" na bazie mastera:

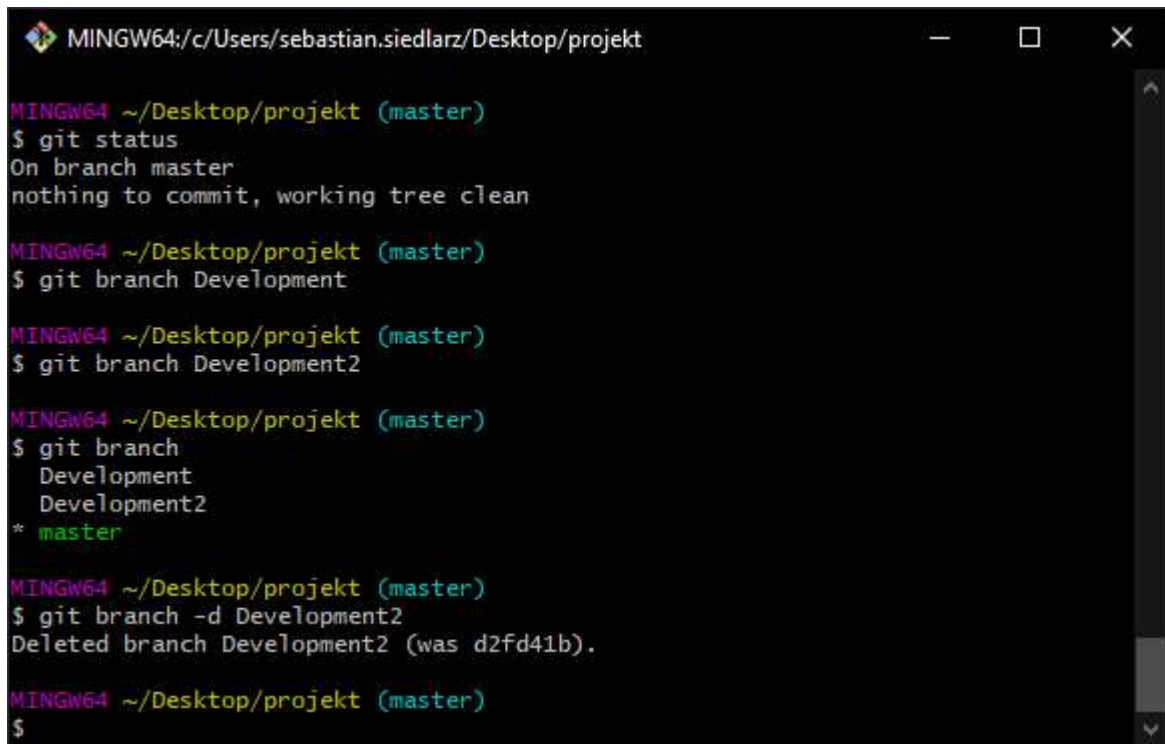
git branch Development

Lista branchów:

git branch

Usuwanie brancha:

git branch -d Development

A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt shows a series of Git commands and their outputs. The prompt is "MINGW64 ~/Desktop/projekt (master)". The first command is "\$ git status", which outputs "On branch master" and "nothing to commit, working tree clean". The second command is "\$ git branch Development", which outputs nothing. The third command is "\$ git branch Development2", which also outputs nothing. The fourth command is "\$ git branch", which outputs "Development", "Development2", and "* master" (with "master" highlighted in green). The fifth command is "\$ git branch -d Development2", which outputs "Deleted branch Development2 (was d2fd41b)". The final command is "\$", which outputs nothing. The window has a vertical scrollbar on the right side.

```
MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 ~/Desktop/projekt (master)
$ git branch Development

MINGW64 ~/Desktop/projekt (master)
$ git branch Development2

MINGW64 ~/Desktop/projekt (master)
$ git branch
  Development
  Development2
* master

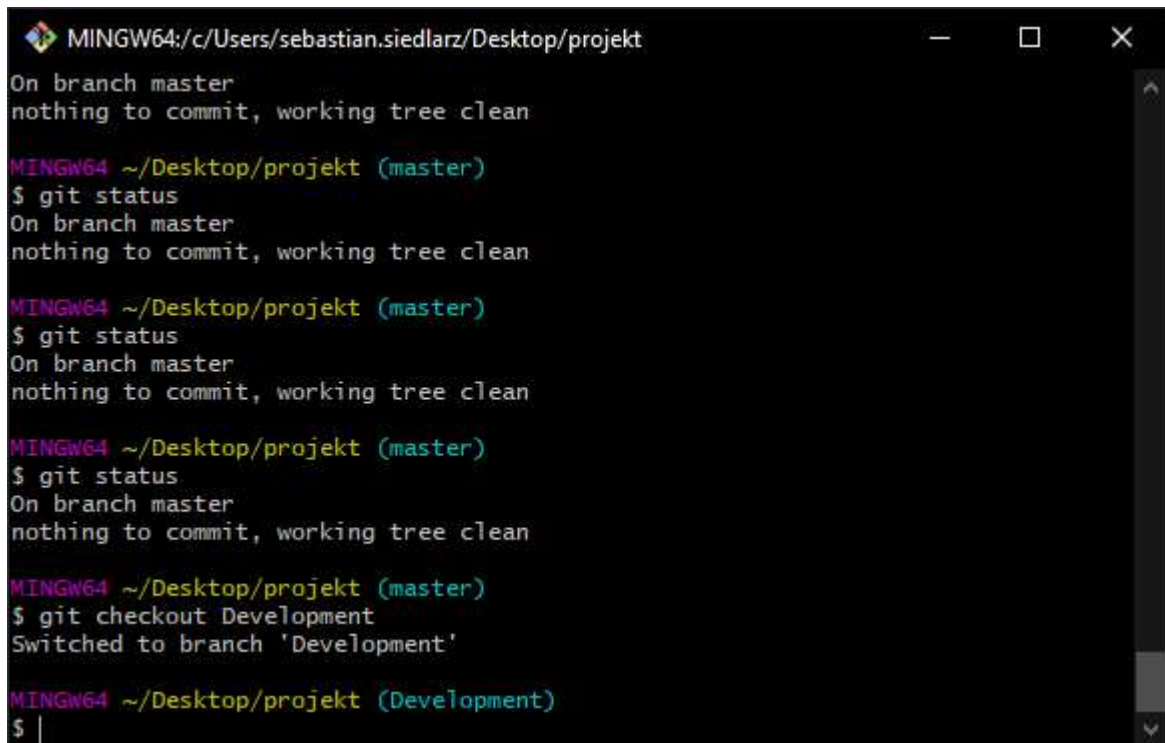
MINGW64 ~/Desktop/projekt (master)
$ git branch -d Development2
Deleted branch Development2 (was d2fd41b).

MINGW64 ~/Desktop/projekt (master)
$
```

Przełączanie gałęzi:

git checkout nazwa -> git checkout Development

Jeżeli mamy niezapisane zmiany (commit) nie pozwoli się przełączyć. Pozostaje albo je usunąć albo wrzucić na "stash" (następny podpunkt).



```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
On branch master
nothing to commit, working tree clean

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 ~/Desktop/projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 ~/Desktop/projekt (master)
$ git checkout Development
Switched to branch 'Development'

MINGW64 ~/Desktop/projekt (Development)
$ |
```

Łączenie gałęzi odbywa się poleceniem:

git merge nazwa_gałęzi -> git merge Development

z poziomu gałęzi, do której wprowadzamy zmiany

Przykład:

Na branchu development dodałem jedną zmianę. Przełączyłem się do branch master i połączyłem go z branchem Development.

Oczywiście branch nie znika i ciągle możemy na nim pracować po czym powtórzyć proces mergowania.

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
$ git checkout Development
Switched to branch 'Development'

MINGW64 ~/Desktop/projekt (Development)
$ git add .

MINGW64 ~/Desktop/projekt (Development)
$ git commit -m "Druga zmienna globalna"
[Development 61aa59f] Druga zmienna globalna
1 file changed, 2 insertions(+)

MINGW64 ~/Desktop/projekt (Development)
$ git checkout master
Switched to branch 'master'

MINGW64 ~/Desktop/projekt (master)
$ git merge Development
Updating d2fd41b..61aa59f
Fast-forward
 1_git/main.c | 2 ++
1 file changed, 2 insertions(+)

MINGW64 ~/Desktop/projekt (master)
$ |
```

Cofanie zmian

git reset:

--soft

Cofamy się do wybranego commitu a zmiany dokonane po nim zostają na "stage" jako zmiany do zapisania

--mixed

Jak wyżej, ale zmiany trzeba będzie dodać na stage (git add)

--hard

Zmiany są całkowicie tracone

git reset jest nie zalecany zwłaszcza gdy pracujemy ze zdalnym repozytorium

Najlepsze zastosowanie to sytuacja gdy chcemy wywalić zmiany od ostatniego commitu:

git reset --hard

git revert

Wykonuje commit cofający wskazany commit.

git revert kod

```
MINGW64:/c/Users/sebastian.siedlarz/Desktop/projekt
4d38c66 Helloworld
1a30a0b Inicjalizacja repozytorium

MINGW64 ~/Desktop/projekt (master)
$ git revert 61aa59f
hint: Waiting for your editor to close the file...
(electron) Sending uncompressed crash reports is deprecated and will be removed
in a future version of Electron. Set { compress: true } to opt-in to the new beh
avior. Crash reports will be uploaded gzipped, which most crash reporting server
s support.
[master 6ff9368] Revert "Druga zmienna globalna"
 1 file changed, 2 deletions(-)

MINGW64 ~/Desktop/projekt (master)
$ git log --oneline
6ff9368 (HEAD -> master) Revert "Druga zmienna globalna"
61aa59f (Development) Druga zmienna globalna
d2fd41b (origin/master) Refaktor
4f12169 Dodanie zmiennej globalnej
4d38c66 Helloworld
1a30a0b Inicjalizacja repozytorium

MINGW64 ~/Desktop/projekt (master)
$ |
```

Schowek

git stash

Po co:

- pozwala cofnąć zmiany do momentu ostatniego commita i umieścić je w schowku dzięki czemu możemy przetestować inne zmiany lub przełączyć branch
- za pomocą schowka można przenosić niezapisane zmiany na inny branch (przez przypadek zaczęliśmy pisać kod nie na tym branchu co chcieliśmy)

git stash pop

Ściąga ostatnio wrzucone zmiany do schowka i umieszcza je w aktualnym workspace

git stash list

Wyświetla indeksowaną listę

stash show stash@{x}

Ściąga dane o konkretnym indeksie do aktualnego workspace