

プログラミング応用

<http://bit.ly/ouyou3d>

ソースコードの管理 (1)

前期 第2週

2018/4/17

バージョン管理とは

ファイルの変更履歴を記録する

主にプログラム
(ソースコード)
が対象









「バージョン」
と呼ぶ

バージョン管理の利点

- ▶ プログラムが動かなくなっても、動いていた状態に戻すことができる
- ▶ ソースコードの配付が容易になる
- ▶ 共同開発者との共同開発が容易になる
 - 現在の商業開発ではバージョン管理システムの導入は必須

従来のバージョン管理は・・・

ファイル名で区別するように工夫するが・・・

名前		
	課題1_20180417.c	
	課題1_20180418.c	
	課題2_改良版.c	
	課題2_確定版.c	
	課題2_最新版.c	
	課題3.c	
	課題3のコピー.c	
	課題3のコピー2.c	

日付を入れて区別できるが、変更内容は不明

結局どれが一番新しいのかが不明

OSが自動で付けるファイル名のまま

バージョン管理システム

リポジトリに変更履歴を残していく



```
#include <stdio.h>

int main(void)
{
    return 0;
}
```

コミット1
(新規追加)

```
#include <stdio.h>

int main(void)
{
    printf("Hello");
    return 0;
}
```

コミット2
(Hello出力)

```
#include <stdio.h>
int add(int x)
{
    return x+10;
}
int main(void)
{
    printf("Hello");
    return 0;
}
```

コミット3
(関数add追加)

リポジトリ

(今回は「prog-st16d??」)

コミットすると変更内容, 変更時間, コメントなどが自動的に記録される

Gitによるバージョン管理

▶ ステージング・・・バージョン管理するファイルを指定

◆基本コマンド

```
$ git add ファイル名
```

◆全てファイルを管理する場合

```
$ git add -A
```

▶ コミット・・・ステージングしたファイルの変更を記録

◆基本コマンド

```
$ git commit -m "コメント"
```

◆コメントは後から見ても分かるような内容にする

例：課題1完了, 関数addの計算処理を修正

Gitによるバージョン管理

▶ プッシュ・・・GitHubにコミット内容を同期

◆基本コマンド (origin masterの意味は次週)

```
$ git push origin master
```

①新規作成時に

add

②変更履歴を残す度に

commit

③GitHubと同期する時に

push

```
#include <stdio.h>

int main(void)
{
    int x;
    printf("Hello");
    return 0;
}
```



リポジトリ

(今回は「prog-st16d??」)

GitHub

今回の演習の前に・・・

- ▶ 前回のGitHubの準備を完了してください
- ▶ できた人は、前回の練習1-1, 練習1-2に進んで下さい

【練習2-1】

前回練習1-2で作成したプログラムで、配列の要素の値を一部変更して、実行を確認してみましょう。
確認できたら、以下のコマンドを実行して、コミットとプッシュを試してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -a -m “配列の要素変更”
```

```
$ git push origin master
```

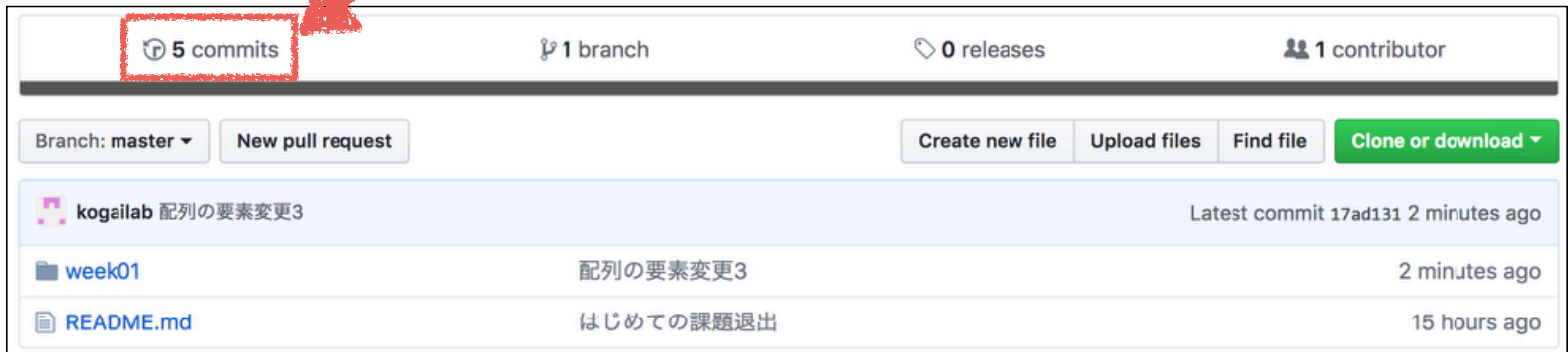
2. 以下の自分の課題用リポジトリを開いて、更新が反映されていることを確認する（次のスライド参考）

[https://github.com/nit-ibaraki-ouyou/prog-\(ユーザー名\)](https://github.com/nit-ibaraki-ouyou/prog-(ユーザー名))

【練習2-1】

自分のリポジトリでコミット内容を確認します

クリック



The screenshot shows the GitHub interface for a repository named 'kogailab'. At the top, there are statistics: '5 commits' (highlighted with a red box and a red arrow pointing to it with the text 'クリック'), '1 branch', '0 releases', and '1 contributor'. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The main content area shows a list of commits. The first commit is titled 'week01' and is described as '配列の要素変更3', committed '2 minutes ago'. The second commit is titled 'README.md' and is described as 'はじめての課題退出', committed '15 hours ago'.

Commit	Message	Time
week01	配列の要素変更3	2 minutes ago
README.md	はじめての課題退出	15 hours ago

【練習2-1】

自分のリポジトリでコミット内容を確認します

クリック

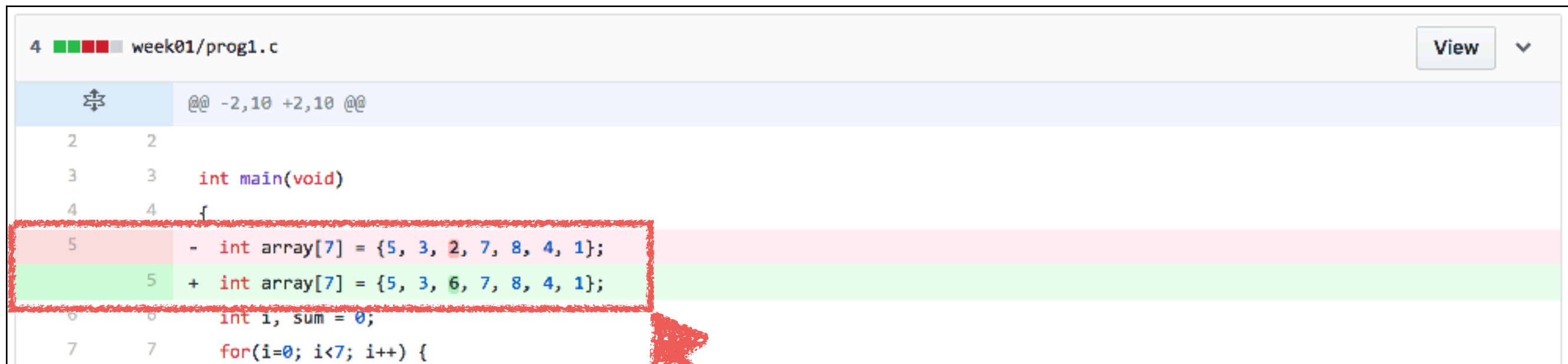


The screenshot shows a GitHub commit history page. A red dashed box highlights the first commit, and a red arrow points to it from the word 'クリック' (Click). The commit list includes:

Commit Message	Author	Time	Commit Hash	Code Diff
配列の要素変更	kogailab	committed 28 minutes ago	d20565d	<>
第1週練習提出	kogailab	committed 38 minutes ago	5563080	<>
Commits on Apr 16, 2018				
はじめての課題退出	kei534	committed 15 hours ago	7926bb4	<>

【練習2-1】

自分のリポジトリでコミット内容を確認します



```
4  ■■■ week01/prog1.c View
@@ -2,10 +2,10 @@
2      2
3      3      int main(void)
4      4      {
5      5      - int array[7] = {5, 3, 2, 7, 8, 4, 1};
6      6      + int array[7] = {5, 3, 6, 7, 8, 4, 1};
7      7      int i, sum = 0;
8      8      for(i=0; i<7; i++) {
```

5行目の変更前 (-) と変更後 (+) の内容が確認できる

【課題2-1】

練習2-1で作成したプログラムに、平均値を計算して出力する処理を追加して、実行を確認してみましょう。確認できたら、以下のコマンドを実行して、**コミットとプッシュ**を試してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -a -m “平均値の処理を追加”
```

```
$ git push origin master
```

2. 以下の自分の課題用リポジトリを開いて、更新が反映されていることを確認する（練習2-1参考）

```
https://github.com/nit-ibaraki-ouyou/prog-\(ユーザ名\)
```

【課題2-2】

前回の練習1-2を参考にして、「week02」というフォルダを作り、その中に「prog2.c」というファイル名で、以下の処理をするプログラムを作ってください。

1. 「prog2.c」に以下のCプログラムを作成する
 - ・ 配列宣言する `int array[7] = {5, 3, 2, 7, 8, 4, 1}`
 - ・ `main()`で、上記の配列の**最大値**を求めて出力する
2. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
$ git commit -m "課題2-2提出"
$ git push origin master
```
3. 自分の課題用リポジトリを開いて、week02がアップロードされているのを確認する
`https://github.com/nit-ibaraki-ouyou/prog-\(ユーザー名\)`