Data Structure #2

2020년 1학기

- 포인터(Pointer)
- 재귀함수(recursive function)

실습 주제 소개

POINTER

포인터(Pointer)

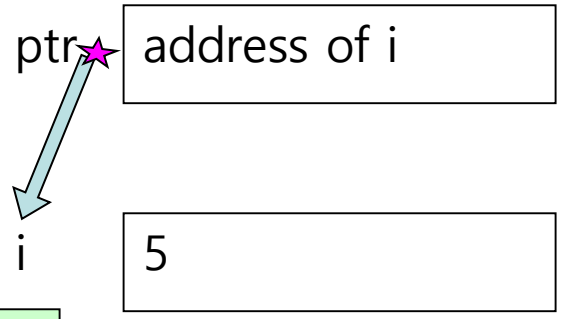
- **포인터**
  - 포인터(pointer)는 프로그래밍 언어에서 다른 변수, 혹은 그 변수의 메모리 공간주소를 가리키는 변수를 말한다.
  - 모든 변수는 메모리에 값을 저장한다. 모든 변수는 메모리에 할당된다. 이러한 메모리의 공간을 구별하는 것이 메모리 주소 값이다.
  - e.g. i = 40; a[2] = 'z';

Addr	Content	Addr	Content	Addr	Content	Addr	Content
1000	i: 40	1001	j: 46	1002	k: 58	1003	m: 74
1004	a[0]: 'a'	1005	a[1]: 'b'	1006	a[2]: 'z'	1007	a[3]: '\0'
1008	ptr: 1001	1009	…	1010		1011	

포인터(Pointer) 사용방법

- 포인터는 주소 값을 담는 변수이다.

```
#include <stdio.h>
int main(){
  int i = 5;
  int *ptr;   /* declare a pointer variable */
  ptr = &i;   /* store address-of i to ptr */
  printf("i = %d\n", i);
  printf("*ptr = %d\n", *ptr);
  printf("ptr = %p\n", ptr);
  return 0;
}
```



Output:

```
i = 5
*ptr = 5
ptr = effff5e0
```

value of ptr = address of i in memory

포인터의 이해

```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```

Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10

포인터의 이해

```
int i = 5, j = 10;
int *ptr;     /* declare a pointer-to-integer variable */
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```

Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	

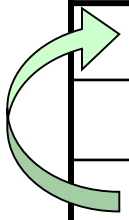
포인터의 이해

```
int i = 5, j = 10;
int *ptr;
int **pptr;  /* declare a pointer-to-pointer-to-integer variable */
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```

Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	
pptr	int **	integer pointer pointer variable	

포인터의 이해

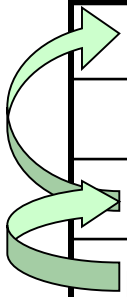
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;     /* store address-of i to ptr */
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	
*ptr	int	de-reference of ptr	5

포인터의 이해

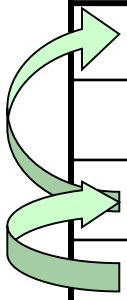
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr; /* store address-of ptr to pptr */
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*pptr	int *	de-reference of pptr	value of ptr (address of i)

포인터의 이해

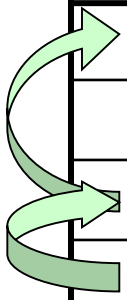
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	3
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*ptr	int	de-reference of ptr	3

포인터의 이해

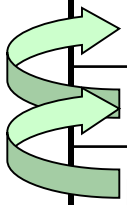
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
**pptr	int	de-reference of de-reference of pptr	7

포인터의 이해

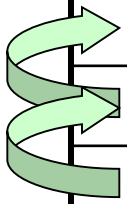
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	10
ptr	int *	integer pointer variable	address of j
pptr	int **	integer pointer pointer variable	address of ptr
*ptr	int	de-reference of ptr	10

포인터의 이해

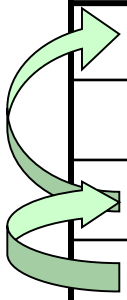
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	9
ptr	int *	integer pointer variable	address of j
pptr	int **	integer pointer pointer variable	address of ptr
**pptr	int	de-reference of de-reference of pptr	9

포인터의 이해

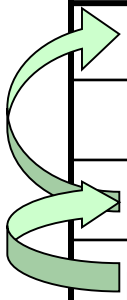
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	9
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*pptr	int *	de-reference of pptr	value of ptr (address of i)

포인터의 이해

```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	-2
j	int	integer variable	9
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*ptr	int	de-reference of ptr	-2

포인터

- 실습내용 코드

```
#include <stdio.h>

int main(void)
{
  int i = 5, j = 10;
  int *ptr;
  int **pptr;
  ptr = &i;
  pptr = &ptr;
  *ptr = 3;
  printf("i: %d, j: %d",i,j);
  **pptr = 7;
  printf("i: %d, j: %d",i,j);
  ptr = &j;
  **pptr = 9;
  printf("i: %d, j: %d",i,j);
  *pptr = &i;
  *ptr = -2;
   printf("i: %d, j: %d",i,j);
  return 0;
}
```

개인 실습 #1

POINTER

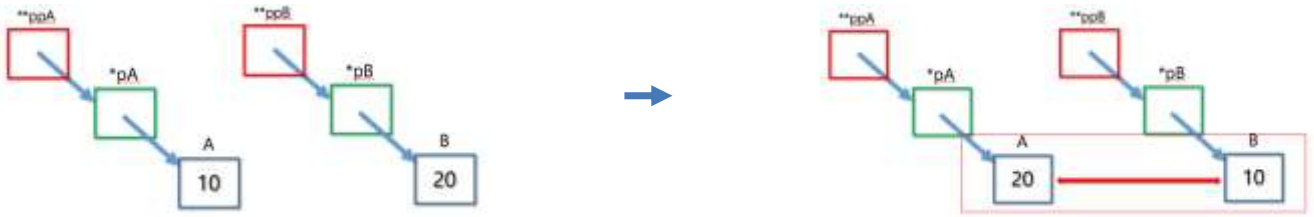
개인 실습 문제 (1/2)

- **Pointer for integers**
  - Obtain 2 integers from the standard input, and swap two integers as follows. Use the concepts of pointer in your work. Print your result on the standard output.

```
variable A : 10, B : 20
After Swap A : 20, B : 10
계속하려면 아무 키나 누르십시
```

개인 실습 문제 (1/2)

- **Pointer for integers (HINT)**




```c
#include <stdio.h>

void swap(int **a, int **b);

int main()
{
    int A=10, B=20;
    int *pA, *pB;

    pA = &A;
    pB = &B;

    printf("Before A = %d\n", *pA);
    printf("Before B = %d\n", *pB);

    swap(&pA, &pB);

    printf("After A = %d\n", *pA);
    printf("After B = %d\n", *pB);
}

void swap(int **a, int **b)
{
    int *temp;

    이 부분을 완성하시오
}
```

실습 주제 소개

Recursive function

재귀함수(Recursive function)

- **재귀함수(**Recursive fuction**)**
  - 재귀(Recursion)는 수학이나 컴퓨터 과학 등에서 자신을 정의할 때 자기 자신을 재참조하는 방법을 뜻한다. 이 방법을 함수에 적용한 형태다.

재귀함수(Recursive function)의 이해

- 재귀 함수의 기본적 이해
  - 자기 자신을 다시 호출하는 형태의 함수

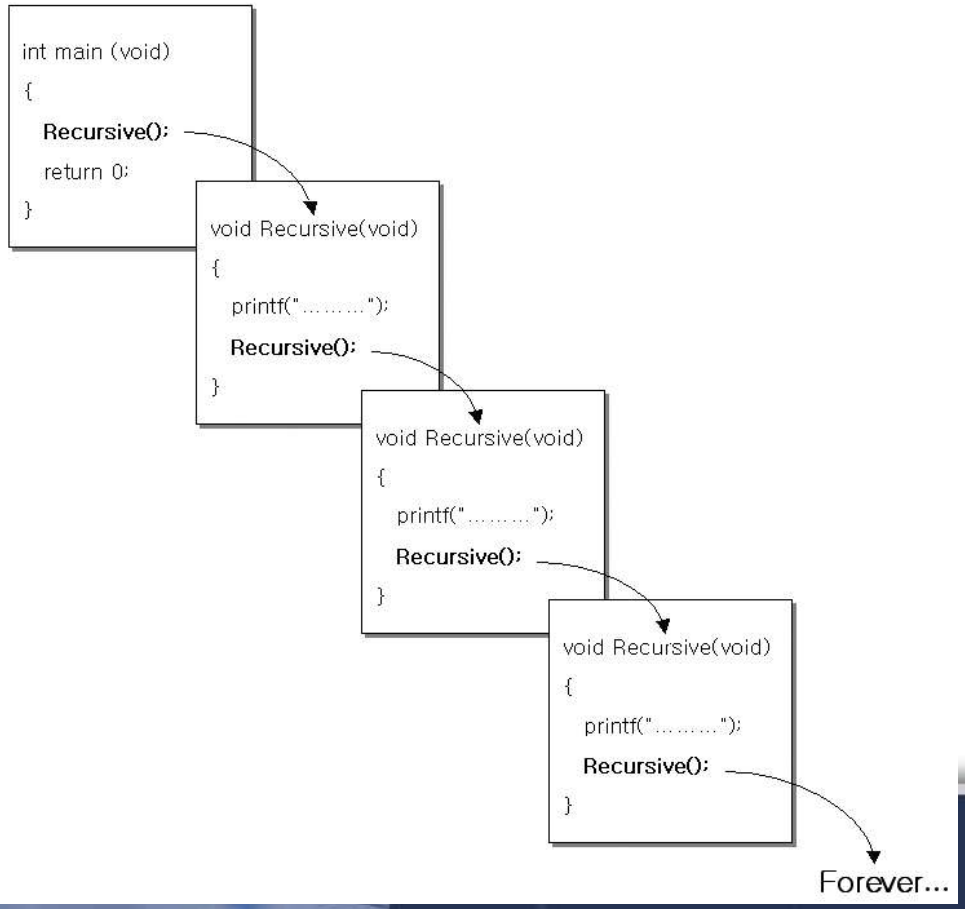
```
/* 무한 반복하며 자기 자신을 호출함*/
#include <stdio.h>

void Recursive(void)
{
   printf("Recursive Call! ₩n");
   Recursive();
}

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

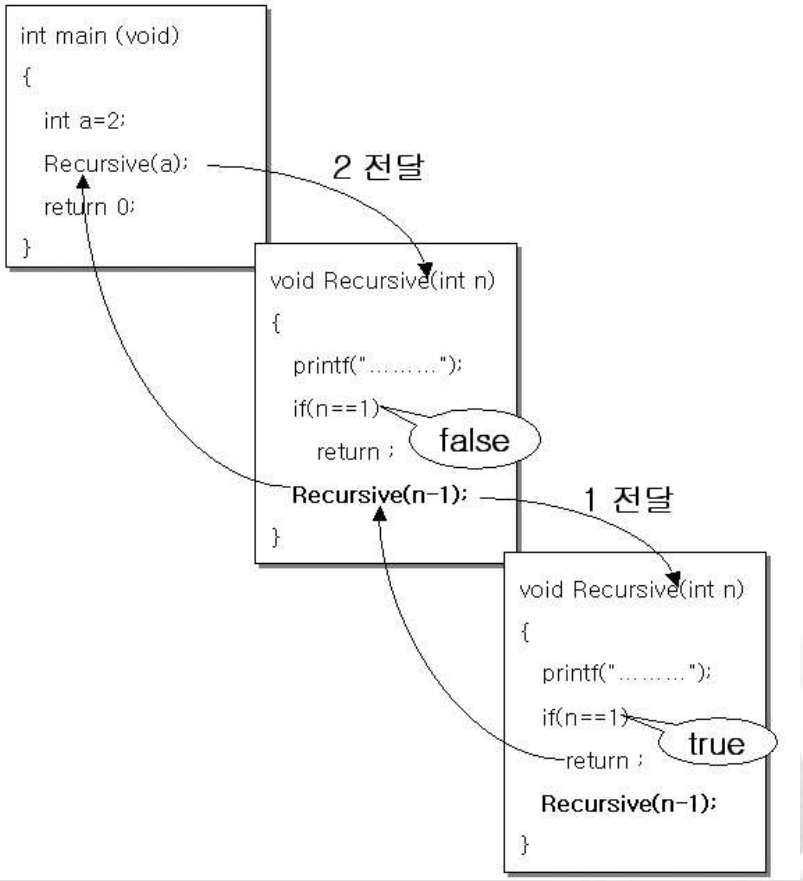
재귀함수(Recursive function)의 이해

- 탈출 조건의 필요성
  - 무한 재귀 호출을 피하기 위해서



재귀함수(Recursive function)의 이해

- 탈출 조건의 이해



재귀함수(Recursive function)의 이해

- **무한루프가 해결된 재귀함수 코드**

```
#include <stdio.h>

void Recursive(int n)
{
    printf("Recursive Call! ₩n");
    if(n==1) return;
    Recursive(n-1);
}

int main(void)
{
    int a=2;
    Recursive(a);
    return 0;
}
```

개인 실습 #2

Recursive function

개인 실습 문제 (2/2)

- **Recursive function**
  - 피보나치 수 구하기

  - input : 자연수 n
  - output : n항까지의 피보나치 수열

```
n = 10
p0 = 0
p1 = 1
p2 = 1
p3 = 2
p4 = 3
p5 = 5
p6 = 8
p7 = 13
p8 = 21
p9 = 34
p10 = 55
```

개인 실습 문제 (2/2)

- **Recursive function(HINT)**

```c
#include <stdio.h>

int main(void){
    int num;

    printf("n = ");
    scanf("%d",&num);

    for(int i = 0; i <=num; i++){
        printf("p%d = %d\n", i, fibonacci(i));
    }

    return 0;
}

int fibonacci(int num){
    switch(num){
    case 0:
        return 0;
    case 1:
        return 1;
    default:
        이 부분을 완성하시오

    }
}
```

Git 기본 사용법안내

1. 설치 완료 후, Git 사용자 설정하기

```
$git config --global user.name “학번”
$git config –global user.email “gitlab에 등록한 email”
```

(user.name은 학번으로,
user.email은 GitLab에 등록해놓은 email로 (기본값: 학번 + @hanyang.ac.kr))

**sudo 명령어는 유닉스 및 유닉스 계열 운영 체제에서 다른 사용자의 보안 권한과 관련된 프로그램을 구동할 수 있게 해주는 프로그램이다.

Git 기본 사용법안내

2. 생성되어 있는 학생의 Git repository clone받기

```
$git clone http://hconnect.hanyang.ac.kr/2020_CSE2010_11786/2020_CSE2010_자신의 학번.git
```

Git clone 주소는 GitLab webpage의 해당 프로젝트 메인화면으로 이동하여 확인 가능

Git 기본 사용법안내

3. git clone 시 요구하는 Username은 학번으로, Password는 GitLab webpage에서 설정한 password로 입력

```
park@park-VirtualBox:~$ git clone http://hconnect.hanyang.ac.kr/2020_CSE2010_11786
/2020_CSE2010_TEST.git
Cloning into '2020_CSE2010_TEST'...
Username for 'https://hconnect.hanyang.ac.kr': 2014004566
Password for 'https://2014004566@hconnect.hanyang.ac.kr':
warning: redirecting to https://hconnect.hanyang.ac.kr/2020_CSE2010_11786/2020_CSE
2010_TEST.git/
warning: You appear to have cloned an empty repository.
```

Git 기본 사용법안내

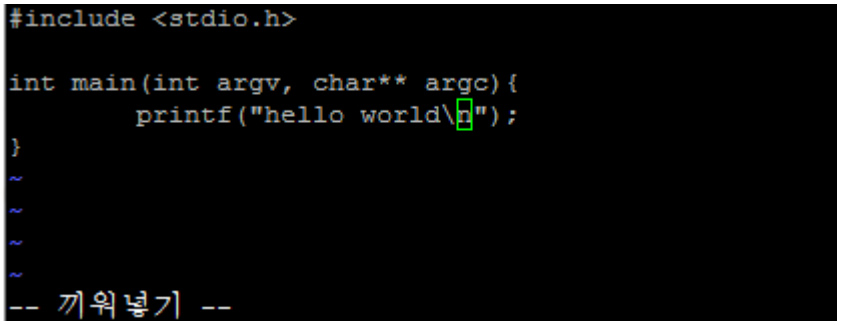
4. Clone받은 폴더로 이동 (처음에는 텅 빈 디렉토리)

```
$cd YEAR_CSE2010_20XXXXXXXX
```

5. 작업 파일 생성

```
$vi test.c
```

6. 파일 작성



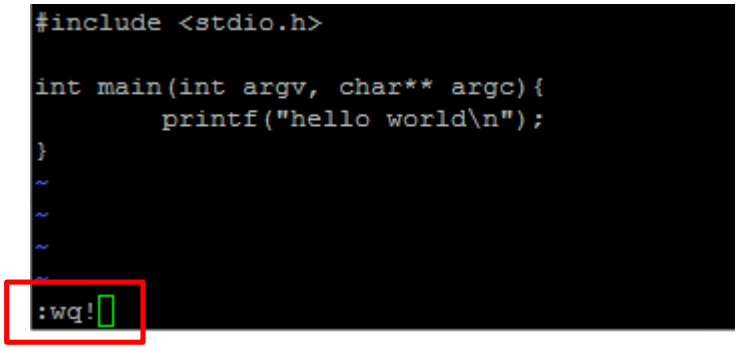

```
#include <stdio.h>

int main(int argv, char** argc){
        printf("hello world₩n");
}
```

Git 기본 사용법안내

7. 파일 저장하기

press [ESC] ► type ":wq"



Git 기본 사용법안내

8. 컴파일 후 실행하기

```
$gcc –o [object file name] [code file name]
```

ex) $sudo gcc –o test.c a.o
$./a.o

```
test@debian:~/data_structure_temp/test$ ./a.o
hello world]ntest@debian:~/data_structure_temp/test$
```

Git 기본 사용법안내

9. 현재 git 관리 상태를 확인하면 test.c가 관리되지 않는 상태로 표시된다.

```
$git status
```

```
park@park-VirtualBox:~/2020_CSE2010_TEST$ git status
On branch master

No commits yet

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

        test.c

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

Git 기본 사용법안내

8. 현재 디렉토리에 있는 모든 추가/수정된 파일들을 Stage 영역으로 추가(test.c가 git에 의해 관리됨)

(. 의 의미는 현재 디렉토리에 있는 모든 파일을 의미한다)

```
$git add .
```

9. Git 관리 상태를 다시 확인

```
$git status
```

```
park@park-VirtualBox:~/2020_CSE2010_TEST$ git add .
park@park-VirtualBox:~/2020_CSE2010_TEST$ git status
On branch master

No commits yet

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

        new file:   test.c
```

Git 기본 사용법안내

10. 추가/수정된 파일을 커밋(Local repository에 저장)

```
$git commit -m “first commit”
```

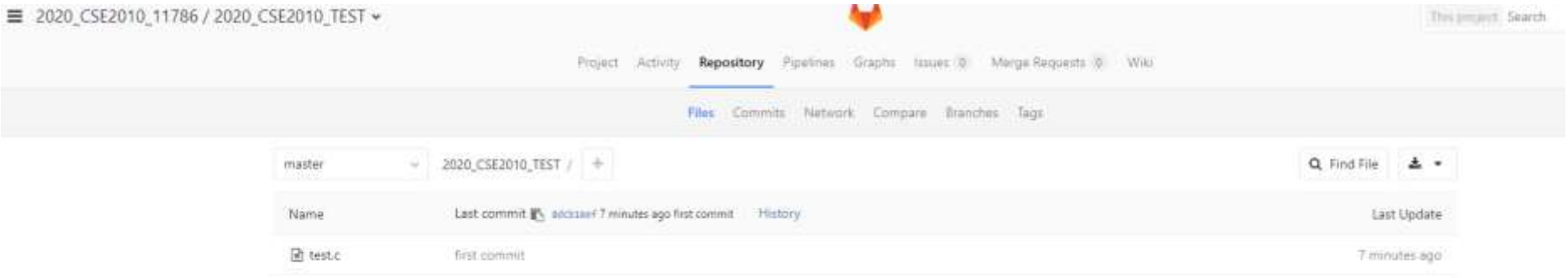
11. 커밋된 내용을 Remote repository로 전송

```
$git push origin master
```

```
park@park-VirtualBox:~/2020_CSE2010_TEST$ git commit -m "first commit"
[master (root-commit) 8dcb16e] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.c
park@park-VirtualBox:~/2020_CSE2010_TEST$ git push origin master
Username for 'https://hconnect.hanyang.ac.kr': 2014004566
Password for 'https://2014004566@hconnect.hanyang.ac.kr':
warning: redirecting to https://hconnect.hanyang.ac.kr/2020_CSE2010_11786/2020_CSE
2010_TEST.git/
Counting objects: 3, done.
Writing objects: 100% (3/3), 208 bytes | 208.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To http://hconnect.hanyang.ac.kr/2020_CSE2010_11786/2020_CSE2010_TEST.git
 * [new branch]      master -> master
```

Git 기본 사용법안내

12. git push를 통해 Remote로 전송된 파일은 GitLab webpage에서 확인 가능하다



제출안내

- **개인실습**
  - 실습 당일 자정까지 제출
- **과제**
  - 다음주 실습 전날 자정까지 제출