


2018 시스템 프로그래밍
- Lab 04 -

제출일자	2018.10.17
분 반	00
이 름	김민기
학 번	201502023

실습 1 - 프로그램 작성


조건 1	student 라는 이름의 struct 정의
------	--------------------------

 struct.c + (~) - VIM

```
1 // 201502023 김민기
2
3 #include <stdio.h>
4
5 struct student {
6     char* name;
7     int number;
8 };
9
```

이름의 이니셜은 문자열 이므로 char 타입의 포인터를 이용해 선언하고, 학번은 숫자이므로 int 타입으로 선언했다.


조건 2	조교와 학생의 student 구조체를 생성 후 정보 입력
------	---------------------------------

 struct.c + (~) - VIM

```
16
17 int main(){
18     struct student me;
19     struct student ta;
20
21     me.name = "kmk";
22     ta.name = "jhs";
23
24     me.number = 201502023;
25     ta.number = 0;
26
```

struct 구조체를 자신과 조교, 2개를 생성하여서 자신의 구조체에는 "kmk"와 201502023을, 조교의 구조체에는 "jhs"와 0의 정보를 저장했다.

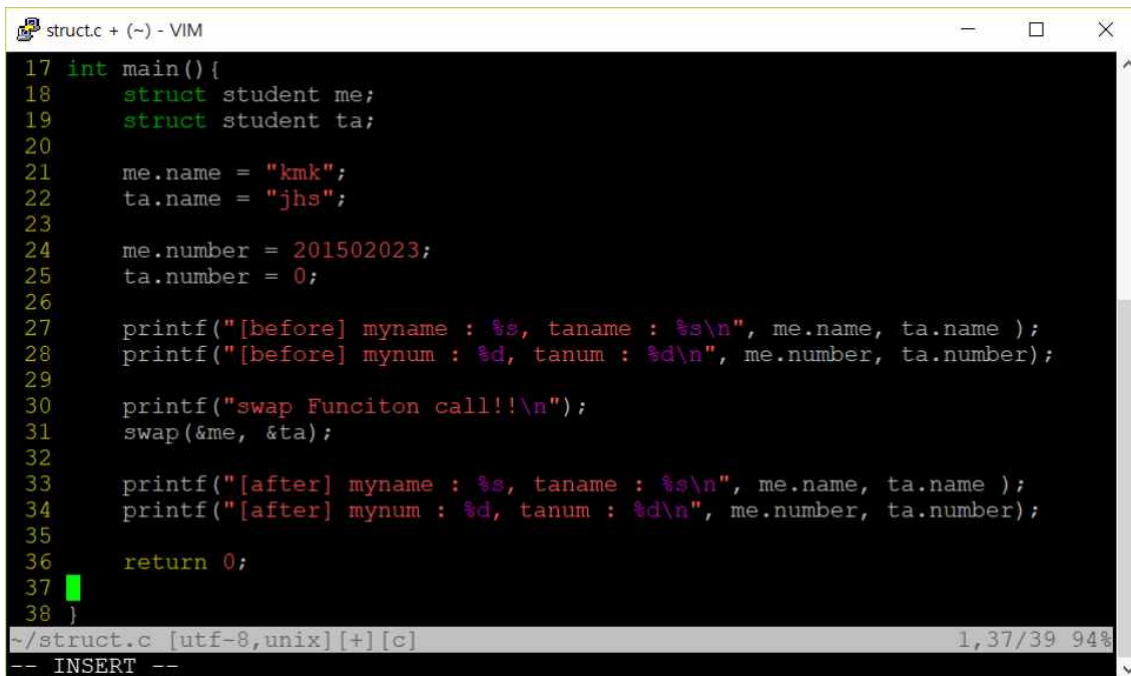
조건 3	swap 함수
------	---------

 struct.c + (~) - VIM

```
10
11 void swap(struct student* arg1, struct student* arg2){
12     struct student temp = *arg1;
13     *arg1 = *arg2;
14     *arg2 = temp;
15 }
16
```

swap함수의 매개변수를 구조체타입의 포인터로 설정해 조교와 나의 정보를 바꿔주었다.

결과화면	셸창에서 결과 화면에 대한 출력
-------------	-------------------



```
17 int main(){
18     struct student me;
19     struct student ta;
20
21     me.name = "kmk";
22     ta.name = "jhs";
23
24     me.number = 201502023;
25     ta.number = 0;
26
27     printf("[before] myname : %s, taname : %s\n", me.name, ta.name );
28     printf("[before] mynum : %d, tanum : %d\n", me.number, ta.number);
29
30     printf("swap Funciton call!!\n");
31     swap(&me, &ta);
32
33     printf("[after] myname : %s, taname : %s\n", me.name, ta.name );
34     printf("[after] mynum : %d, tanum : %d\n", me.number, ta.number);
35
36     return 0;
37
38 }
```

~/struct.c [utf-8,unix][+][c] 1,37/39 94%
-- INSERT --

```
a201502023@2018-sp: ~  
login as: a201502023  
a201502023@133.186.153.97's password:  
Access denied  
a201502023@133.186.153.97's password:  
Access denied  
a201502023@133.186.153.97's password:  
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-130-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
a201502023@2018-sp:~$ ls  
a.out          datalab-handout.tar  operation_test.tar.gz  
datalab-handout operation_test        struct.c  
a201502023@2018-sp:~$ vi struct.c  
a201502023@2018-sp:~$ ./a.out  
[before] myname : kmk, taname : jhs  
[before] mynum  : 201502023, tanum : 0  
swap Funciton call!!  
[after] myname  : jhs, taname : kmk  
[after] mynum   : 0, tanum  : 201502023  
a201502023@2018-sp:~$
```

실습 2

1 학생, 조교 저장 데이터 확인

```
a201502023@2018-sp: ~  
Type "apropos word" to search for commands related to "word"...  
Reading symbols from a.out...done.  
(gdb) b main  
Breakpoint 1 at 0x400629: file struct.c, line 17.  
(gdb) run  
Starting program: /home/sys00/a201502023/a.out  
  
Breakpoint 1, main () at struct.c:17  
17      int main(){  
(gdb) info locals  
me = {name = 0x0, number = 0}  
ta = {name = 0x4006f0 <__libc_csu_init> "AWAVA\211\377AUATL\215%\016\a ",  
      number = 4195552}  
(gdb) n  
21          me.name = "kmk";  
(gdb) n  
22          ta.name = "jhs";  
(gdb) n  
24          me.number = 201502023;  
(gdb) n  
25          ta.number = 0;  
(gdb) n  
27          printf("[before] myname : %s, taname : %s\n", me.name, ta.name )  
;  
(gdb) n  
[before] myname : kmk, taname : jhs  
28          printf("[before] mynum : %d, tanum : %d\n", me.number, ta.number  
);  
(gdb) n  
[before] mynum : 201502023, tanum : 0  
30          printf("swap Funciton call!!\n");  
(gdb) █
```

'gdb a.out' 명령어를 사용해 gdb를 실행했다. main함수에 Break Poing를 걸고 gdb에서 프로그램을 실행했다. 그런 다음 main함수에서 변수에 값이 잘 들어갔는지를 확인하기 위해 'info locals' 명령어를 사용해 자신과 조교가 들어갔는지를 확인했다. n을 통해 한줄씩 넘기면서 swap함수가 실행되기 전까지를 보면 값이 잘 들어간 것을 볼 수 있다.

2 Swap 과정 설명

```
a201502023@2018-sp: ~
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...done.
(gdb) b struct.c: 30
Breakpoint 1 at 0x400687: file struct.c, line 30.
(gdb) run
Starting program: /home/sys00/a201502023/a.out
[before] myname : kmk, taname : jhs
[before] mynum : 201502023, tanum : 0

Breakpoint 1, main () at struct.c:30
30         printf("swap Funciton call!!\n");
(gdb) s
swap Funciton call!!
31         swap(&me, &ta);
(gdb) s
swap (arg1=0x7fffffff510, arg2=0x7fffffff520) at struct.c:12
12         struct student temp = *arg1;
(gdb) display *arg1
1: *arg1 = {name = 0x400778 "kmk", number = 201502023}
(gdb) display *arg2
2: *arg2 = {name = 0x40077c "jhs", number = 0}
(gdb) s
13         *arg1 = *arg2;
1: *arg1 = {name = 0x400778 "kmk", number = 201502023}
2: *arg2 = {name = 0x40077c "jhs", number = 0}
(gdb) s
14         *arg2 = temp;
1: *arg1 = {name = 0x40077c "jhs", number = 0}
2: *arg2 = {name = 0x40077c "jhs", number = 0}
(gdb) s
15     }
1: *arg1 = {name = 0x40077c "jhs", number = 0}
2: *arg2 = {name = 0x400778 "kmk", number = 201502023}
(gdb) s
main () at struct.c:33
33         printf("[after] myname : %s, taname : %s\n", me.name, ta.name );
(gdb) s
[after] myname : jhs, taname : kmk
34         printf("[after] mynum : %d, tanum : %d\n", me.number, ta.number);
(gdb) s
[after] mynum : 0, tanum : 201502023
36         return 0;
(gdb) s
38     }
```

gdb a.out' 명령어를 사용해 gdb를 실행했다. swap함수가 실행되는 줄이 31번째 줄
이므로 30번째 줄에 Break Point를 걸었다. s를 통해 swap함수 내부로 들어가서
display를 이용해 *arg1와 *arg2의 변수값을 매번 확인해준다. 다음줄로 넘어가면서
수값을 확인해보면 자신과 조교의 정보가 바뀐 것을 볼 수 있다.
