TI DSP, MCU, Xilinx Zynq FPGA 기반의 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 김형주
mihaelkel@naver.com

What I leanred (18.03.02)

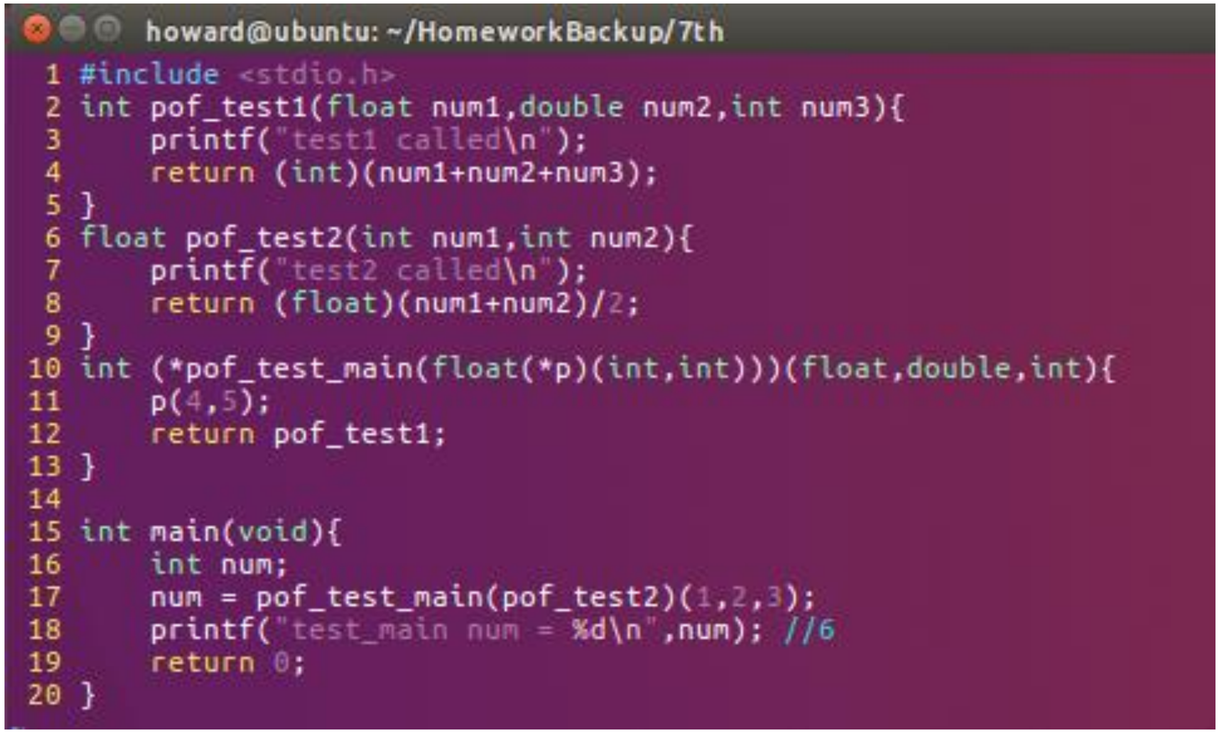
- The reasons why we use function pointer
- Quiz for function pointer
- Data Structure : stack

Why we use function pointer?

- We use function pointer to make asynchronous handling. If You open Linux OS code, you'll see a lot of System call. But these are in 'kernel space', not in 'user space'. So you can't access system call, but you can by using function pointer. For example, writing code for avr system, you can use interrupt function in kernel space.

Quiz for function pointer

- Quiz 1.

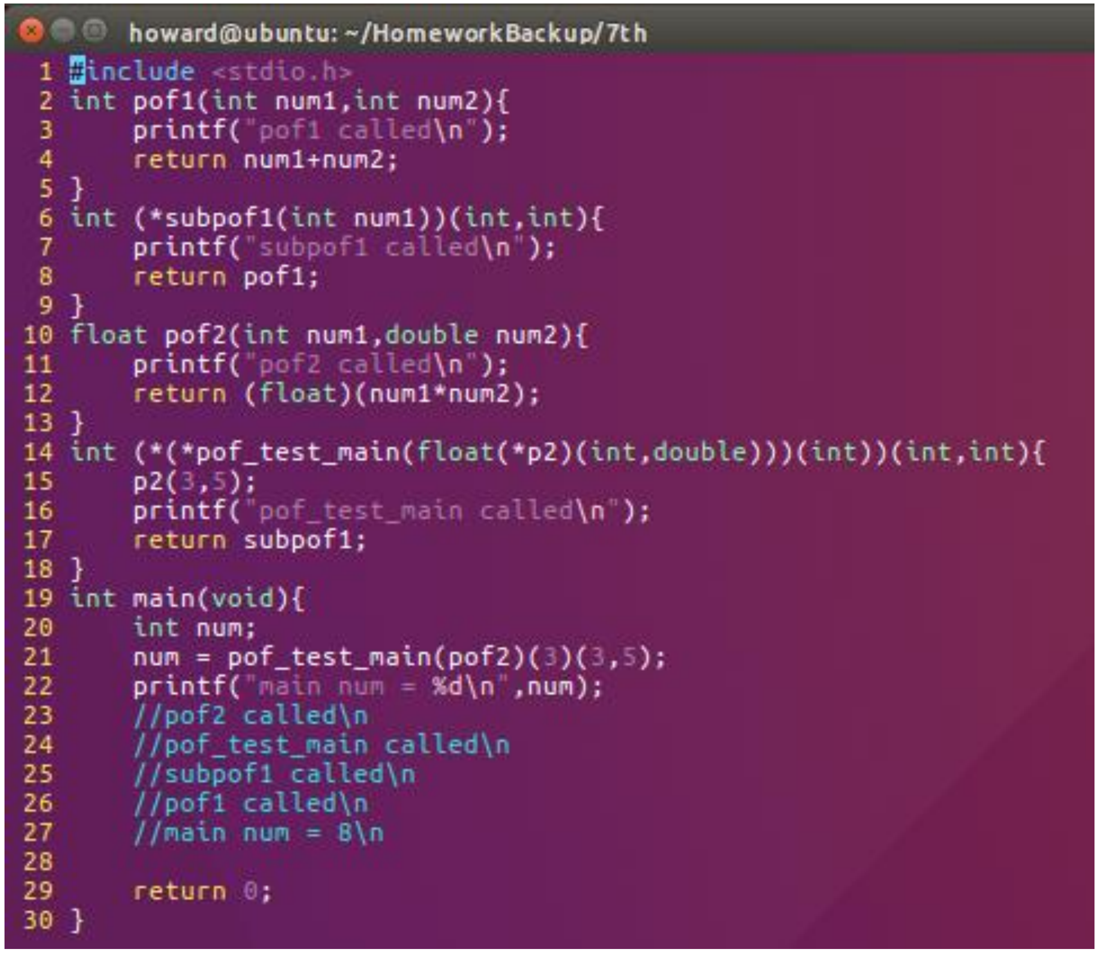


howard@ubuntu: ~/HomeworkBackup/7th

```c
#include <stdio.h>
int pof_test1(float num1,double num2,int num3){
    printf("test1 called\n");
    return (int)(num1+num2+num3);
}
float pof_test2(int num1,int num2){
    printf("test2 called\n");
    return (float)(num1+num2)/2;
}
int (*pof_test_main(float(*p)(int,int)))(float,double,int){
    p(4,5);
    return pof_test1;
}

int main(void){
    int num;
    num = pof_test_main(pof_test2)(1,2,3);
    printf("test_main num = %d\n",num); //6
    return 0;
}
```

Quiz for function pointer

- Quiz 2.

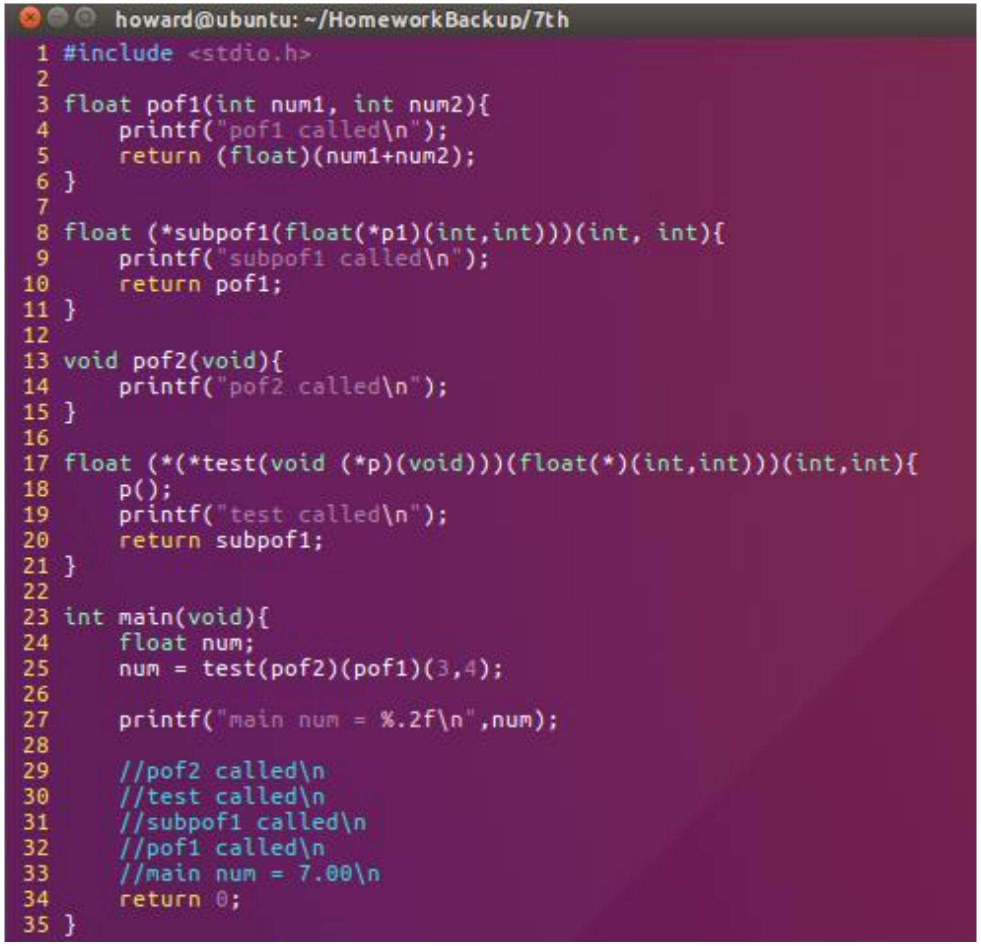


howard@ubuntu: ~/HomeworkBackup/7th

```c
#include <stdio.h>
int pof1(int num1,int num2){
    printf("pof1 called\n");
    return num1+num2;
}
int (*subpof1(int num1))(int,int){
    printf("subpof1 called\n");
    return pof1;
}
float pof2(int num1,double num2){
    printf("pof2 called\n");
    return (float)(num1*num2);
}
int (*(*pof_test_main(float(*p2)(int,double)))(int))(int,int){
    p2(3,5);
    printf("pof_test_main called\n");
    return subpof1;
}
int main(void){
    int num;
    num = pof_test_main(pof2)(3)(3,5);
    printf("main num = %d\n",num);
    //pof2 called\n
    //pof_test_main called\n
    //subpof1 called\n
    //pof1 called\n
    //main num = 8\n

    return 0;
}
```

Quiz for function pointer

- Quiz 3.

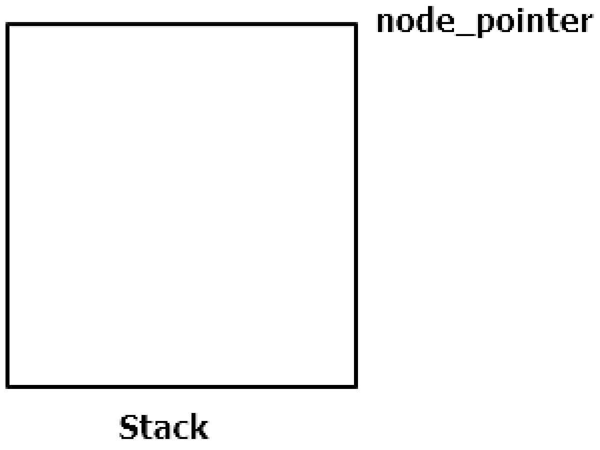


howard@ubuntu: ~/HomeworkBackup/7th

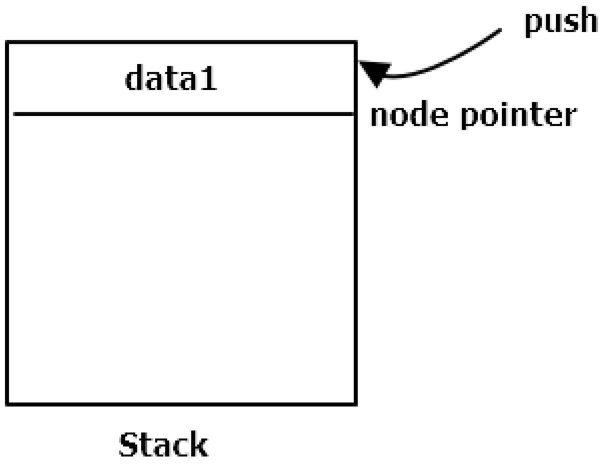
```c
#include <stdio.h>

float pof1(int num1, int num2){
    printf("pof1 called\n");
    return (float)(num1+num2);
}

float (*subpof1(float(*p1)(int,int)))(int, int){
    printf("subpof1 called\n");
    return pof1;
}

void pof2(void){
    printf("pof2 called\n");
}

float (*(*test(void (*p)(void)))(float(*)(int,int)))(int,int){
    p();
    printf("test called\n");
    return subpof1;
}

int main(void){
    float num;
    num = test(pof2)(pof1)(3,4);

    printf("main num = %.2f\n",num);

    //pof2 called\n
    //test called\n
    //subpof1 called\n
    //pof1 called\n
    //main num = 7.00\n
    return 0;
}
```

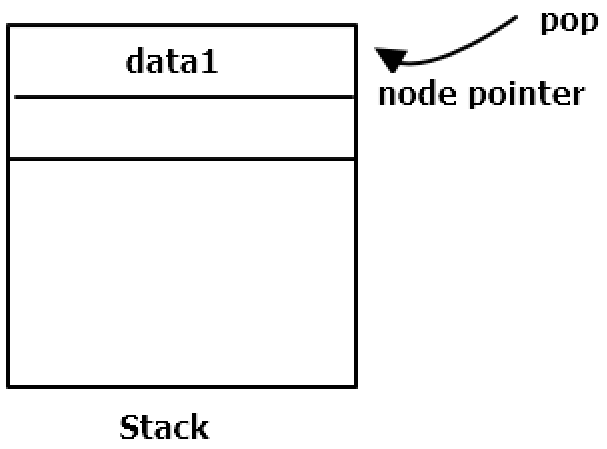
Data Structure : Stack

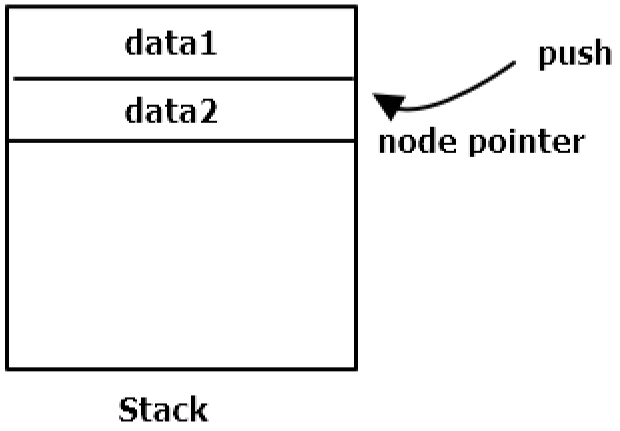
- Stack : FILO structure
- Methode : push, pop



Data Structure : Stack

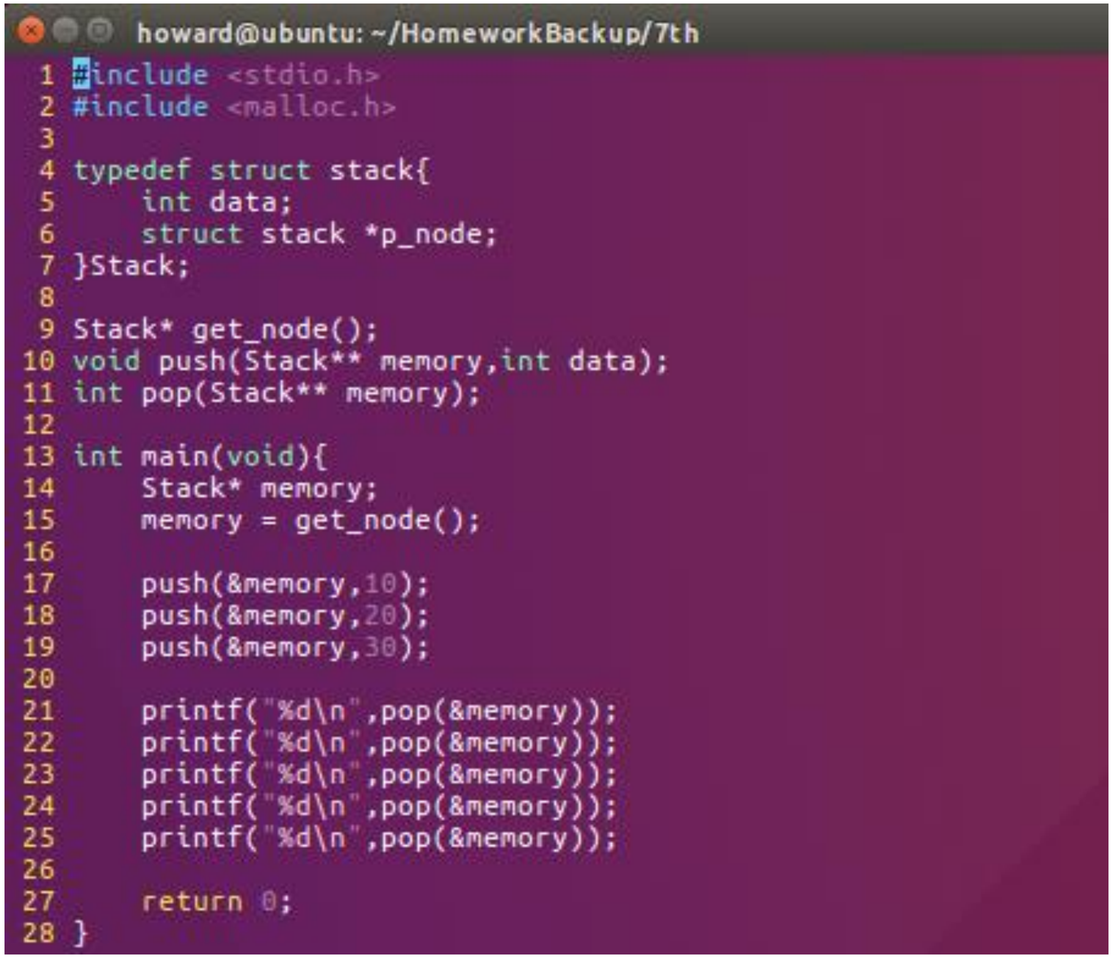






Data Structure : Stack

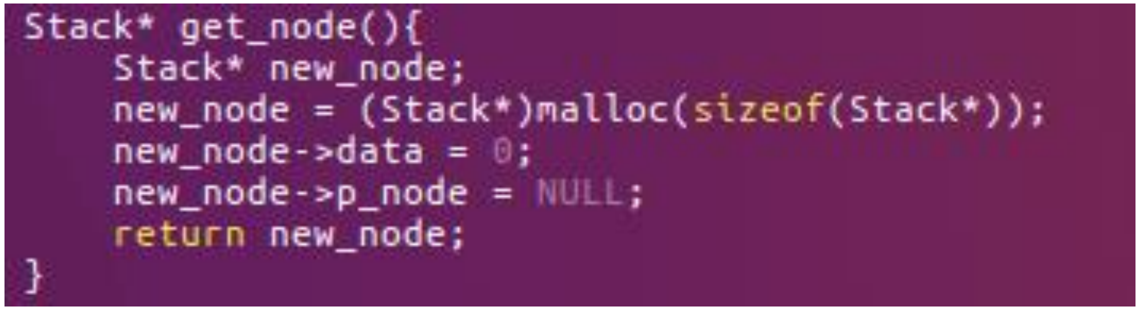
- Consist of main(), get_node(), pop(), push()



```c
#include <stdio.h>
#include <malloc.h>

typedef struct stack{
    int data;
    struct stack *p_node;
}Stack;

Stack* get_node();
void push(Stack** memory,int data);
int pop(Stack** memory);

int main(void){
    Stack* memory;
    memory = get_node();

    push(&memory,10);
    push(&memory,20);
    push(&memory,30);

    printf("%d\n",pop(&memory));
    printf("%d\n",pop(&memory));
    printf("%d\n",pop(&memory));
    printf("%d\n",pop(&memory));
    printf("%d\n",pop(&memory));

    return 0;
}
```

Data Structure : Stack

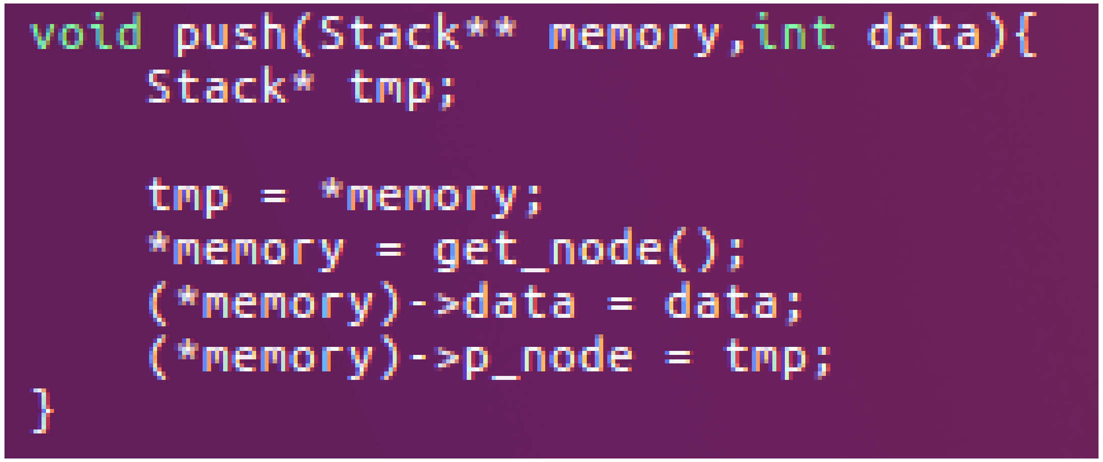
- Stack* get_node()
- This fuction returns Stack type pointer initialized data = 0, p_node = NULL



```
Stack* get_node(){
    Stack* new_node;
    new_node = (Stack*)malloc(sizeof(Stack*));
    new_node->data = 0;
    new_node->p_node = NULL;
    return new_node;
}
```

Data Structure : Stack

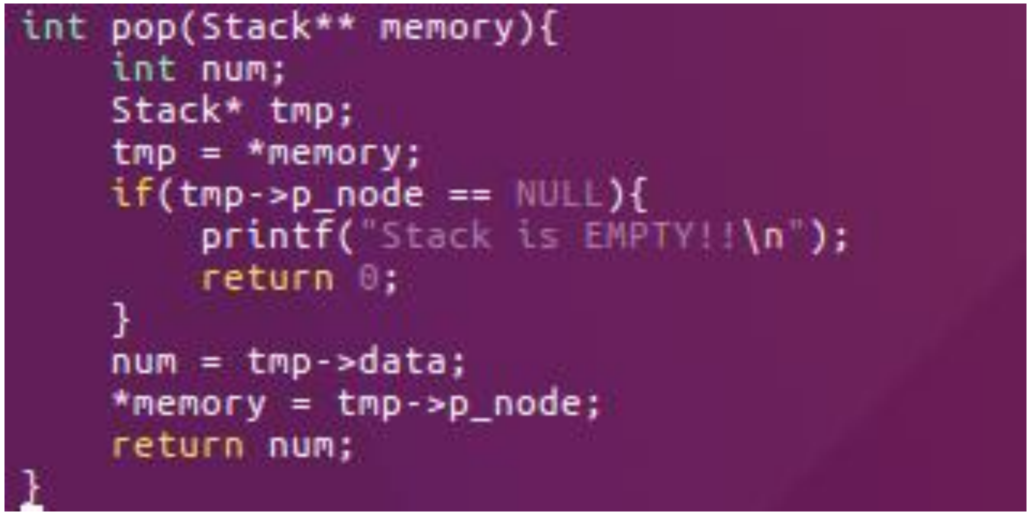
- void push(Stack** memory, int data)
- This fuction accumulates data on memory. Current memory address becomes new p_node address.



```
void push(Stack** memory,int data){
    Stack* tmp;

    tmp = *memory;
    *memory = get_node();
    (*memory)->data = data;
    (*memory)->p_node = tmp;
}
```

Data Structure : Stack

- void pop(Stack** memory)
- This fuction returns data on memory. p_node address become new memory address.
- If there's no data in stack(p_node == NULL), warnning



```
int pop(Stack** memory){
    int num;
    Stack* tmp;
    tmp = *memory;
    if(tmp->p_node == NULL){
        printf("Stack is EMPTY!!\n");
        return 0;
    }
    num = tmp->data;
    *memory = tmp->p_node;
    return num;
}
```