

# Stack 자료 구조

## 구현 및 분석

작성자 : Lee Daero(skseofh@naver.com)

## <자료 구조 Stack 구현 및 분석>

### [목적]

- stack 구조를 그림을 통해 분석한다.

### [stack]

- First In Last Out 의 특징을 가지는 자료구조
- 구현에서는 Linked List 형태로 구현한다.

#### (1) node 구조

#### Stack Node

```
typedef struct __stack
{
    int data;
    struct __stack *link;
} stack;
```

int data
struct __stack link

- data를 담는 int형 변수
- 다음 node를 가리키는 struct \_\_stack \* 형 link 포인터

#### (2) 구현 함수

```
stack *get_stack_node(void);
void push(stack **top, int data);
int pop(stack **top);
void print_stack(stack *top);
```

- 1) get\_stack\_node : 동적할당으로 stack 구조체를 할당 받아서 포인터를 반환
- 2) push : stack에 최상단에 data를 저장
- 3) pop : stack에 최상단의 data를 pop하고 pop한 데이터를 반환
- 4) print\_stack : stack에 저장된 data를 출력

#### (3) 동작 설명

- 10, 20, 30, 40을 stack에 저장하고 출력한다.
- 2번 pop()함수를 호출하여 pop 되는 데이터를 출력하고 stack에 남아있는 data를 출력한다.

```
int main(void)
{
    int i, data;
    stack *top = NULL;

    for(i = 1; i <= 4; i++)
        push(&top, i * 10);

    print_stack(top);

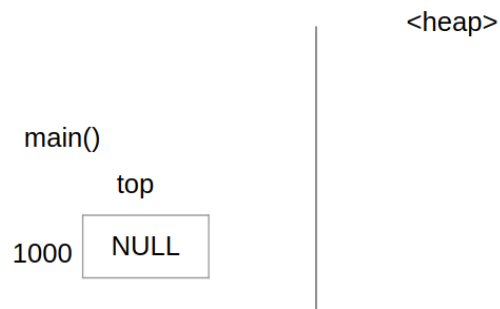
    for(i = 0; i < 2; i++)
        printf("delete data = %d\n", pop(&top));

    print_stack(top);

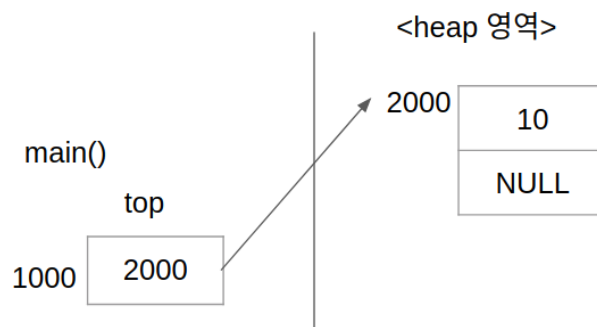
    return 0;
}
```

```
tmp->data = 40
tmp->data = 30
tmp->data = 20
tmp->data = 10
delete data = 40
delete data = 30
tmp->data = 20
tmp->data = 10
```

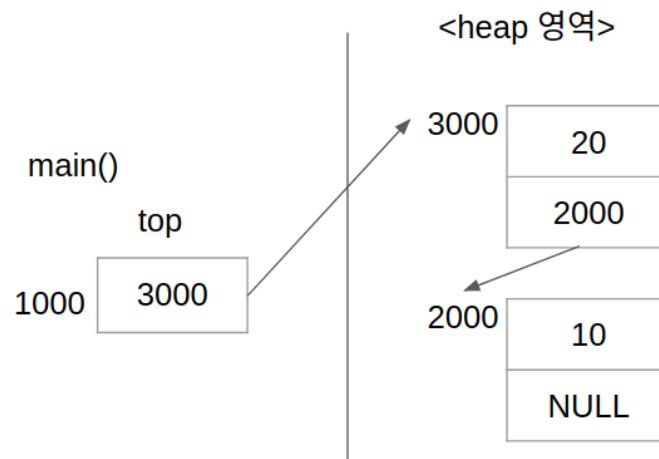
1) stack \* top이 NULL로 초기화 되었을 때



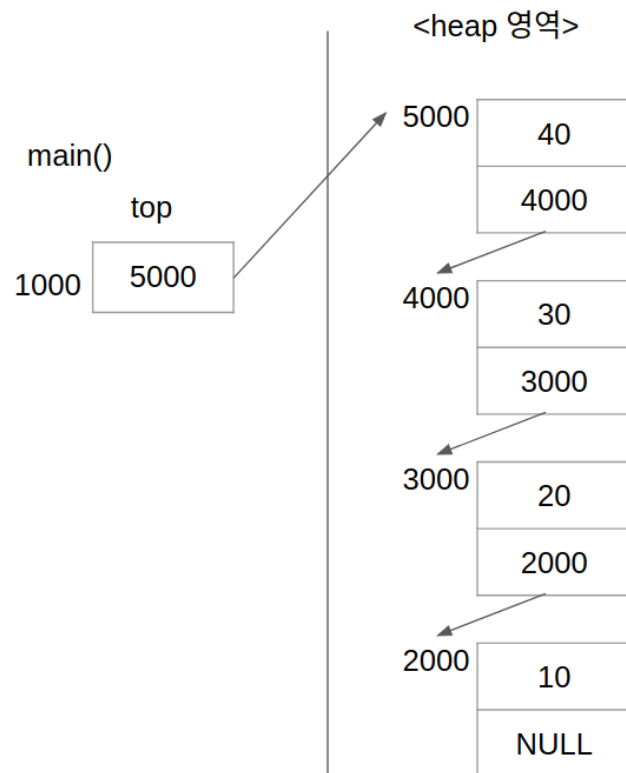
2) 10, 20, 30, 40 순서로 push를 진행  
- 10 push()



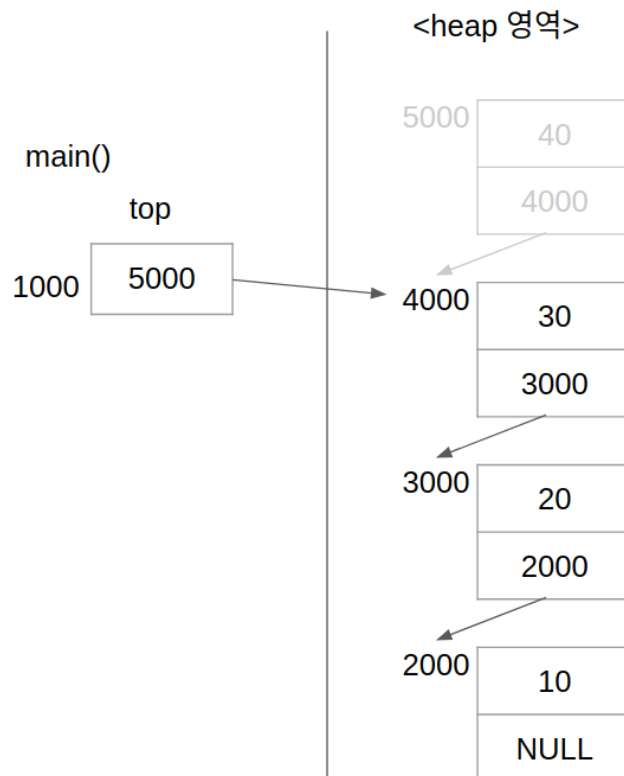
-20 push()



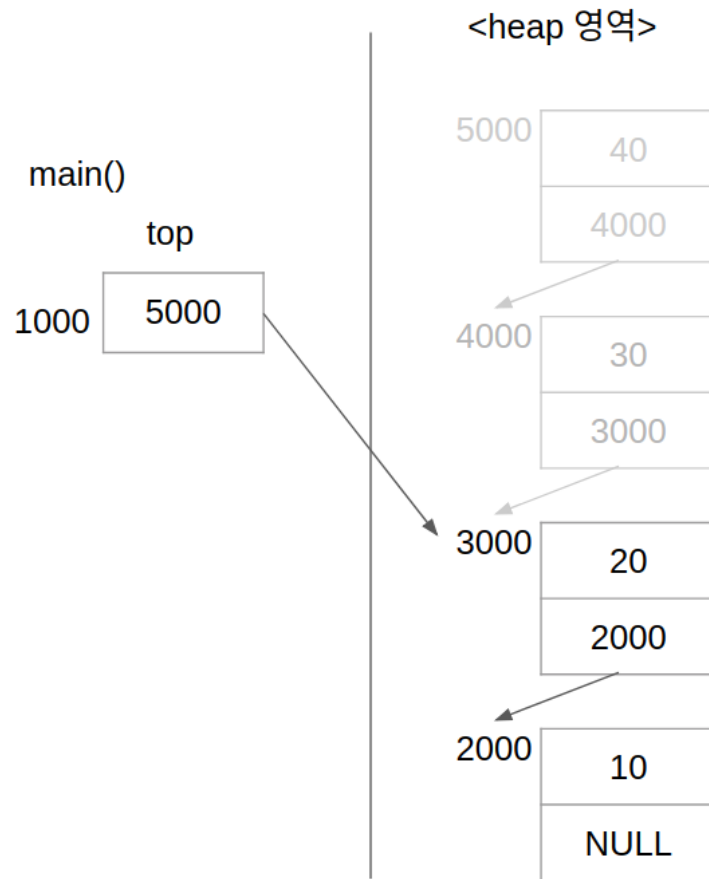
-30, 40 push()



- pop() 1번째



- pop() 2번째



### (3) 함수 분석

1) get\_stack\_node() 함수 : 동적할당으로 stack 구조체를 할당 받아서 포인터를 반환

```
stack *get_stack_node(void)
{
    stack *tmp = (stack *)malloc(sizeof(stack));
    tmp->link = NULL;

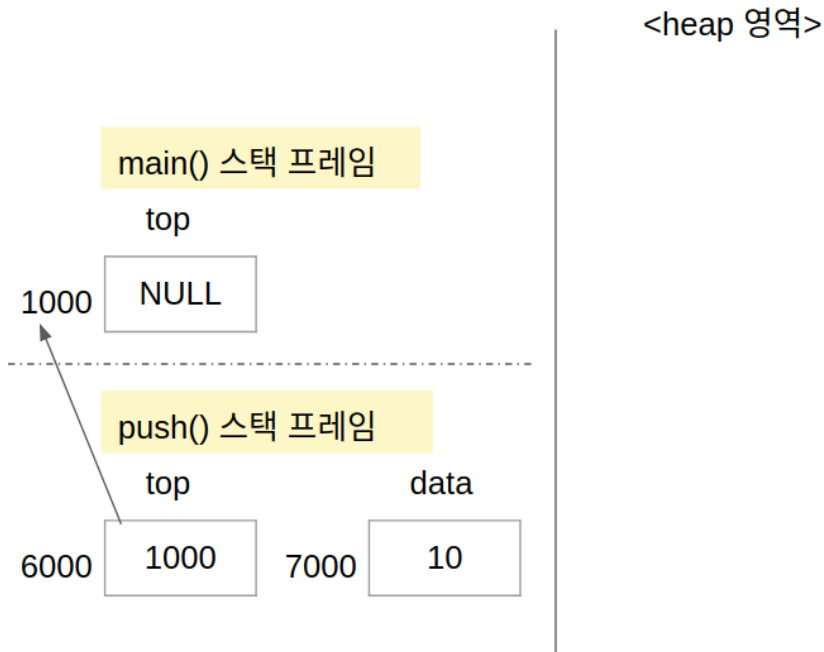
    return tmp;
}
```

- malloc 함수로 stack 구조체 만큼을 stack 포인터(tmp)에 동적할당
- stack 포인터 tmp의 link를 NULL로 초기화
- tmp를 반환

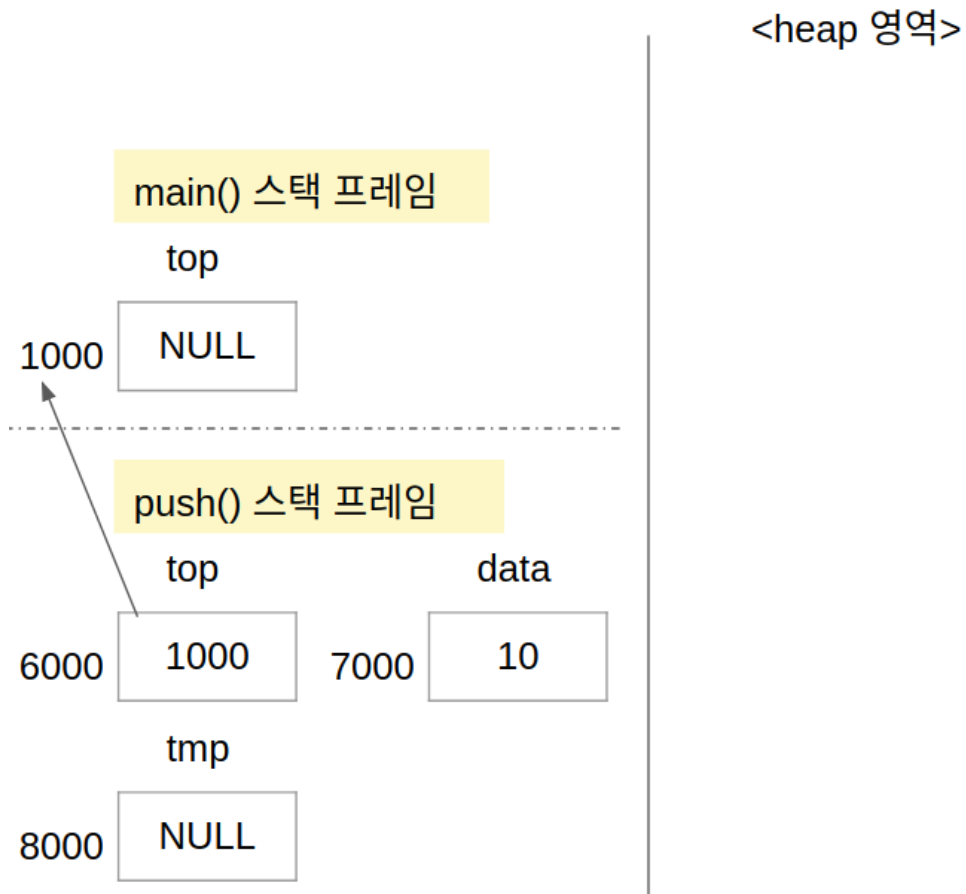
2) push() 함수 : stack에 최상단에 data를 저장

```
void push(stack **top, int data)
{
    stack *tmp = *top;
    *top = get_stack_node();
    (*top)->data = data;
    (*top)->link = tmp;
}
```

- push() 함수로 넘어 왔을 때 메모리 그림
- : push() 스택 프레임에서 stack \*\* top 이 main() 스택 프레임의 top 변수의 주소로 초기화 되어 생성됨
- : int형인 변수가 data로 초기화 되어 생성.



- `stack * tmp = *top` 동작
- : `stack *` 형의 `tmp`를 선언하고 `main()` 함수 `top`의 내용을 저장
- : `push()` 함수의 `top` – `push()` 함수에서 `main()` 함수의 `top`에 접근하여 새로 생성된 노드를 연결하는 역할
- `tmp` – `main()`함수의 `top`의 내용을 저장하기 위한 역할

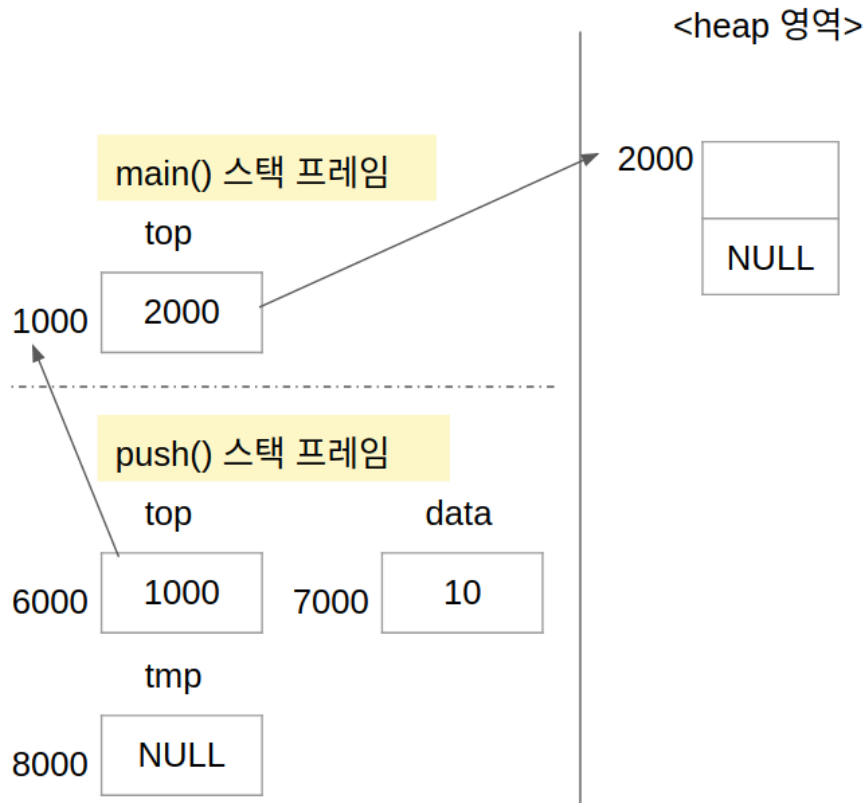


## stack 구조체의 double pointer( `stack ** top`)를 인자로 사용하는 이유

- `main()`함수와 `push()`함수는 스택 프레임이 각각 생성, `push()` 함수가 끝나면 `push()` 함수의 스택 프레임은 없어진다.
- `push()` 함수 내에서 call by reference로 `main()`함수의 `stack * top`에 접근하여 stack 자료 구조를 생성하기 위함.
- `main()` 함수의 `stack * top`이 `stack *` 자료형이므로 call by reference를 하기 위해서는 `stack *`의 주소를 저장할 수 있는 `stack **`를 매개 변수로 넘겨주어야 한다.

- \*top = get\_stack\_node()

: stack 노드 구조체를 할당하여 push() 함수의 top의 주소 접근(\*top)을 통해 main() 함수의 top이 생성된 stack 노드 구조체를 가리키도록 한다.



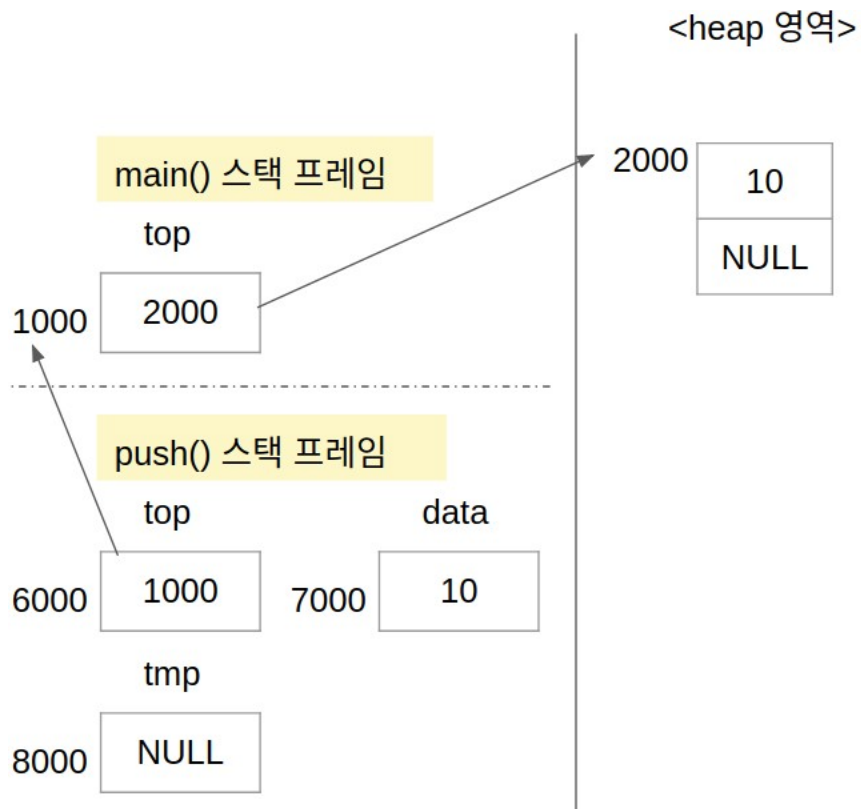
- (\*top) → data = data

: 인자로 넘어온 data 삽입

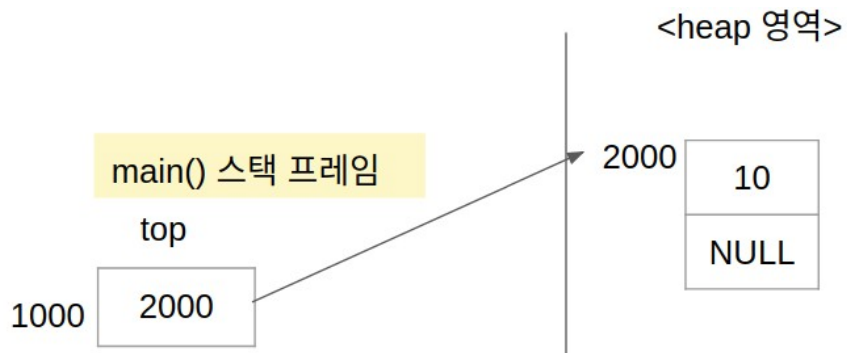
- (\*top) → link = tmp

: 새로 생성된 노드의 link에 tmp의 값, 원래 main()의 top이 가지고 있던 값을 대입하여 stack 노드 연결



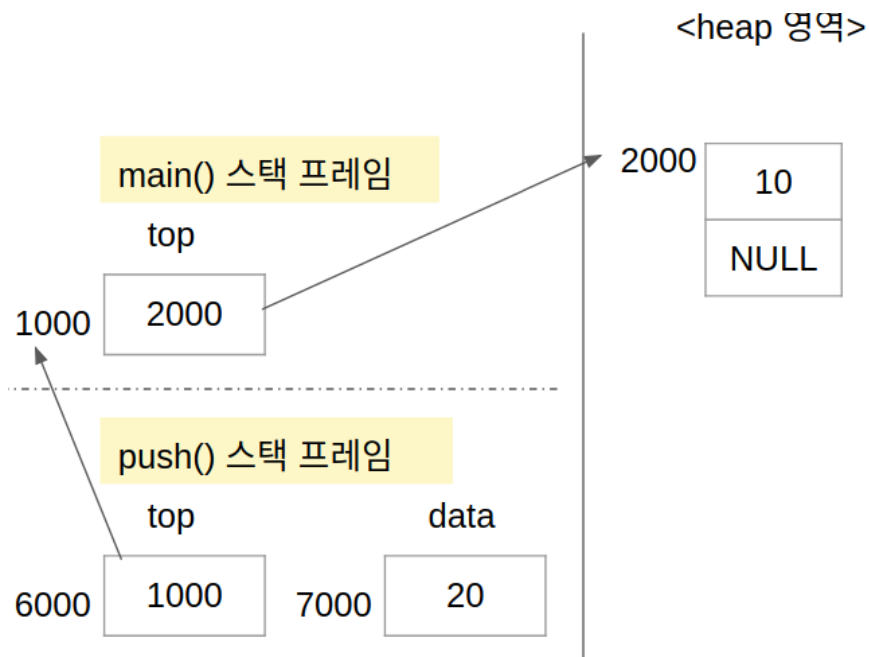


- 함수가 끝나면 push() 스택 프레임이 사라진다. Stack 자료 구조 생성

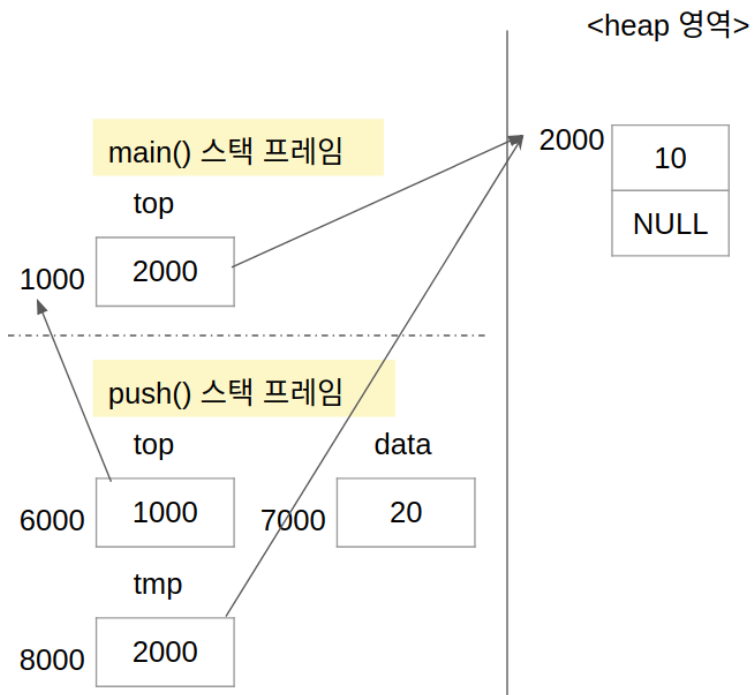


\*\* 20 push 시 push() 함수 스택 프레임 변화 관찰 \*\*

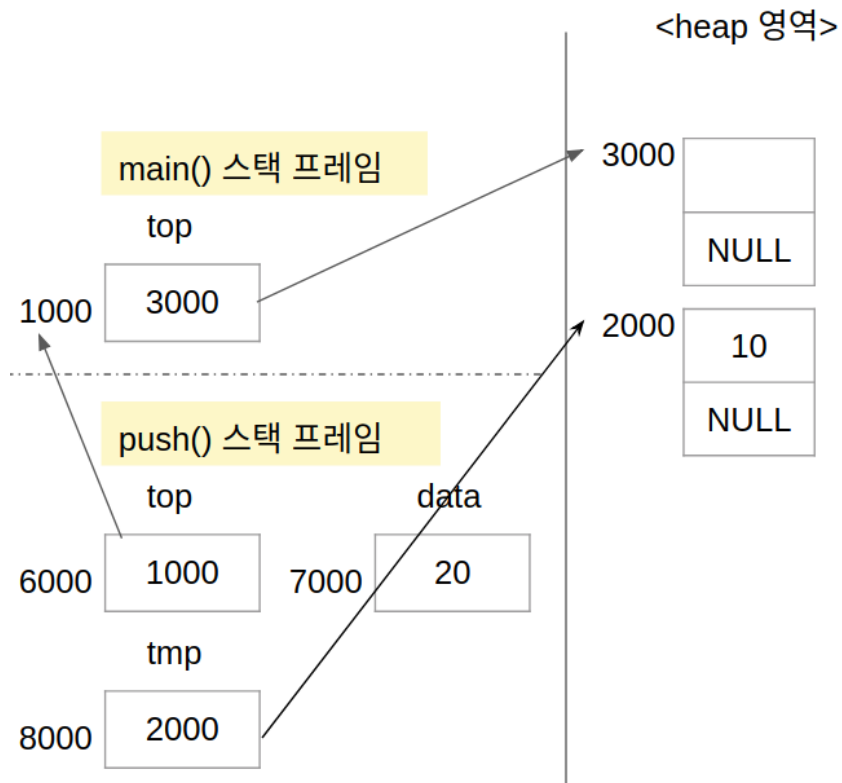
- push() 함수 진입 시 스택 프레임 생성



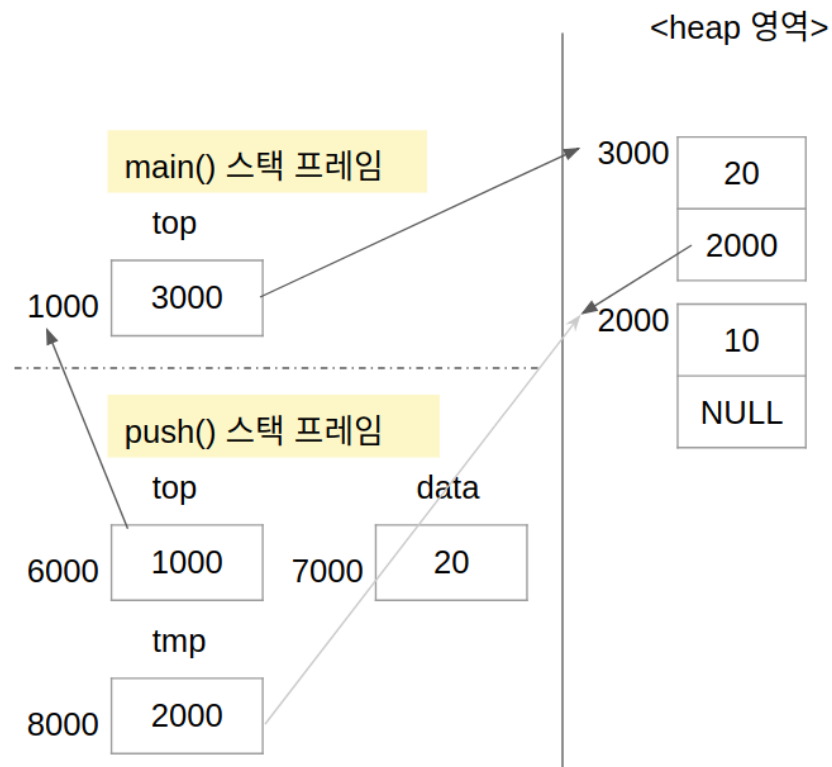
- stack \* tmp = \*top



- \*top = get\_stack\_node()



- (\*top) → data = data
- (\*top) → link = tmp

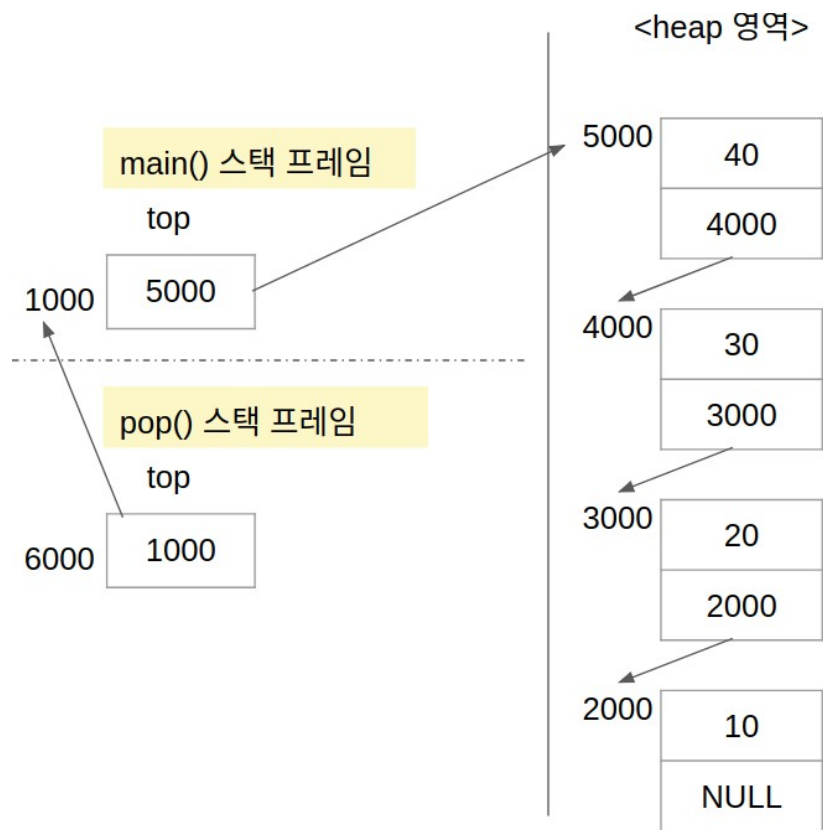


3) `pop()` 함수 : stack에 최상단의 data를 pop하고 pop한 데이터를 반환

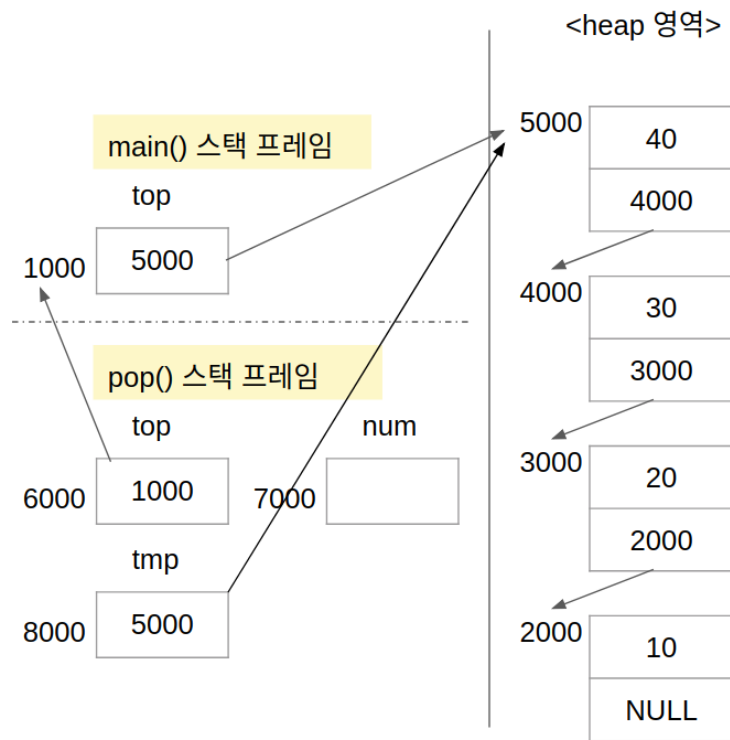
```
int pop(stack **top)
{
    int num;
    stack *tmp = *top;
    if(!tmp)
    {
        printf("stack이 비었습니다\n");
        return 0;
    }
    num = (*top)->data;
    *top = (*top)->link;
    free(tmp);

    return num;
}
```

- 10, 20, 30, 40이 순서대로 stack에 push되었다고 가정.
- `pop()` 함수로 넘어왔을 시의 스택 프레임



- int num
- stack \* tmp = \*top



- if 문

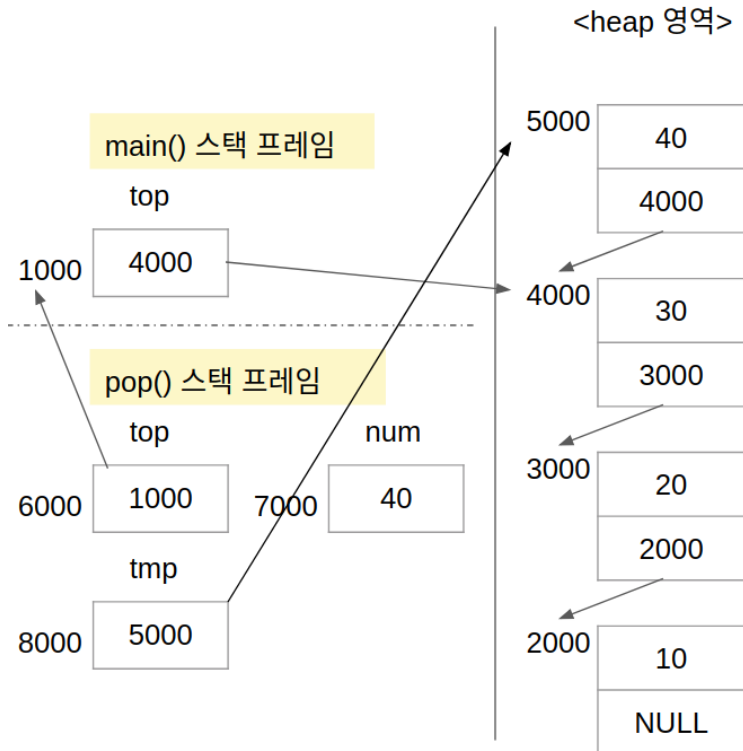
: stack이 비었을 시 return

- num = (\*top) → data

: num에 pop할 데이터 저장

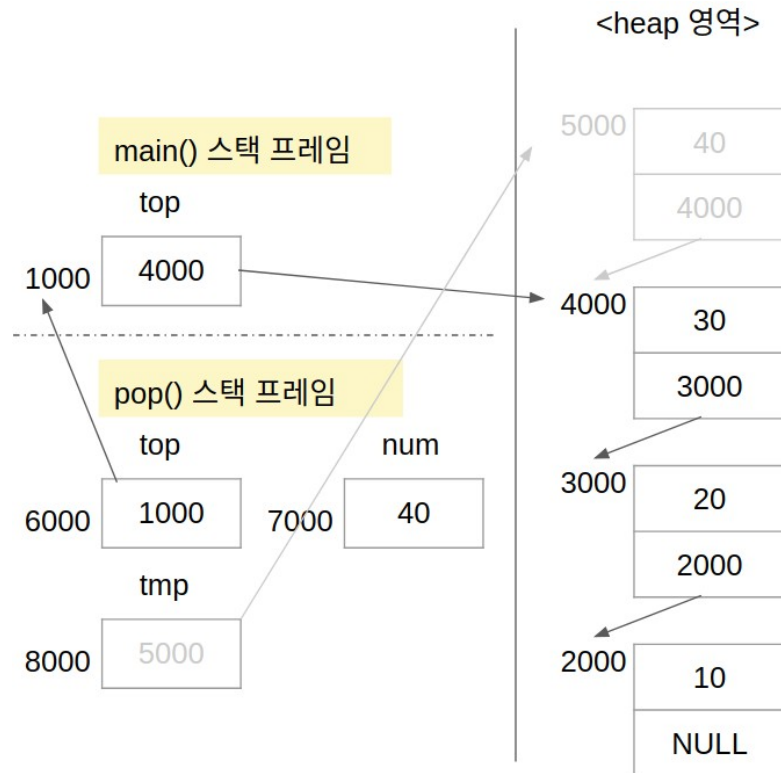
- (\*top) = (\*top) → link

: push() 함수의 top의 주소 접근을 통해 main() 함수의 top이 stack의 최상위에서 하나 다음 것을 가리키게 함



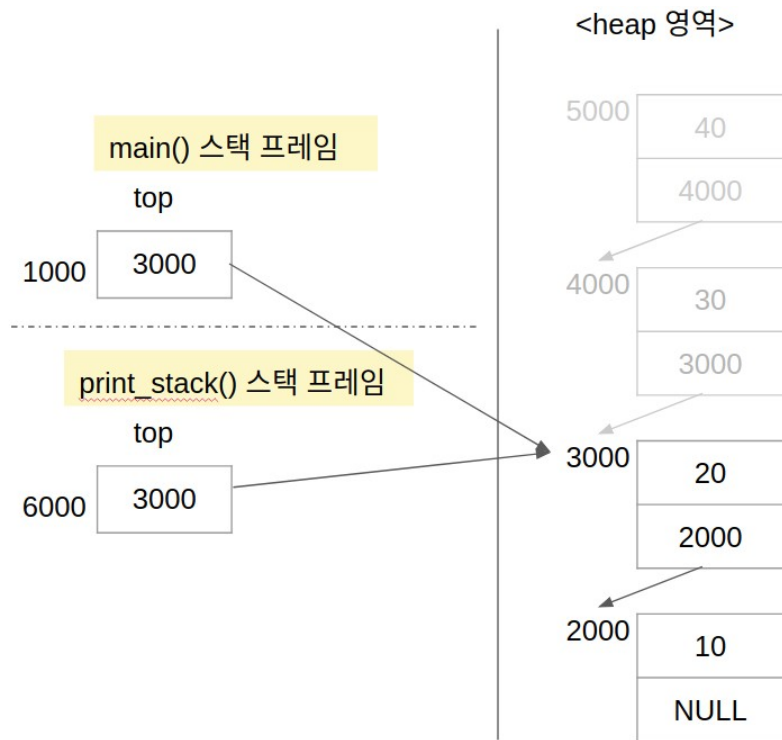
-free(tmp)

: tmp가 가리키던 stack 노드 구조체를 free함  
 - return num  
 : pop한 데이터를 반환



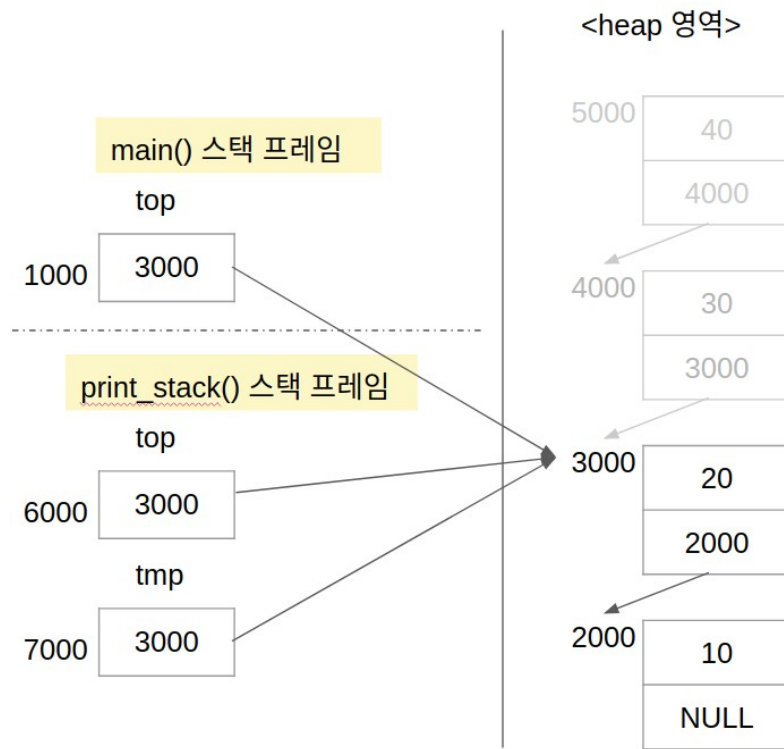
4) `print_stack()` 함수 : stack에 저장된 data를 출력

- 10, 20, 30, 40 push 후 2번 pop하여 10, 20만 stack에 저장되어 있는 상황
- print\_stack() 함수 호출 시



- `stack * tmp = top`





- while문

: tmp가 NULL이 아닐 때까지 link를 따라가면서 data 출력

