TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 문한나
mhn97@naver.com

6일차 내용 정리

- typedef
  자료형에 새로운 이름을 부여하고자 할 때 사용
  주로 구조체나 함수 포인터에 사용함

- malloc()
  Memory 구조상 heap에 data를 할당함
  data가 계속해서 들어올 경우 얼만큼의 data가 들어오는지 알 수 없음
  들어올 때마다 동적으로 할당할 필요성이 있음

- free()
  Memory 구조상 heap에 data를 할당 해제함
  malloc()의 반대 역할을 수행함

반환값은 이 값을 받는 자료형의 포인터로 변환하여
포인터 변수에 저장된다

```
mhn@mhn-900X3L: ~/my_proj/c/6_s
#include <stdlib.h>
#include <stdio.h>

int main(void){

        char *str_ptr = (char *)malloc(sizeof(char) * 20);

        printf("input String : ");
        scanf("%s",str_ptr);

        if(str_ptr != NULL)
                printf("string = %s\n",str_ptr);

        free(str_ptr);

                return 0;

}
```

함수malloc( )의 인자는 할당할 변수의 크기를
sizeof연산자를 이용하여 지정한다

```
mhn@mhn-900X3L:~/my_proj/c/6_s$ ./a.out
input String : hello
string = hello
mhn@mhn-900X3L:~/my_proj/c/6_s$
```

함수free( )는 인자로 해체할 메모리 공간의
주소값을 갖는 포인터를 이용하여 호출한다

- calloc()

  malloc()과 완전히 동일함

  사용 방법에 차이가 있음 1번째 인자는 할당할 개수, 2번째 인자는 할당할 크기

  즉, calloc(2, sizeof(int))는 8byte 공간을 할당

- 구조체

  서로 다른 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형

  문자열과, 숫자를 한 번에 묶어서 관리하고 싶을 때 등 사용

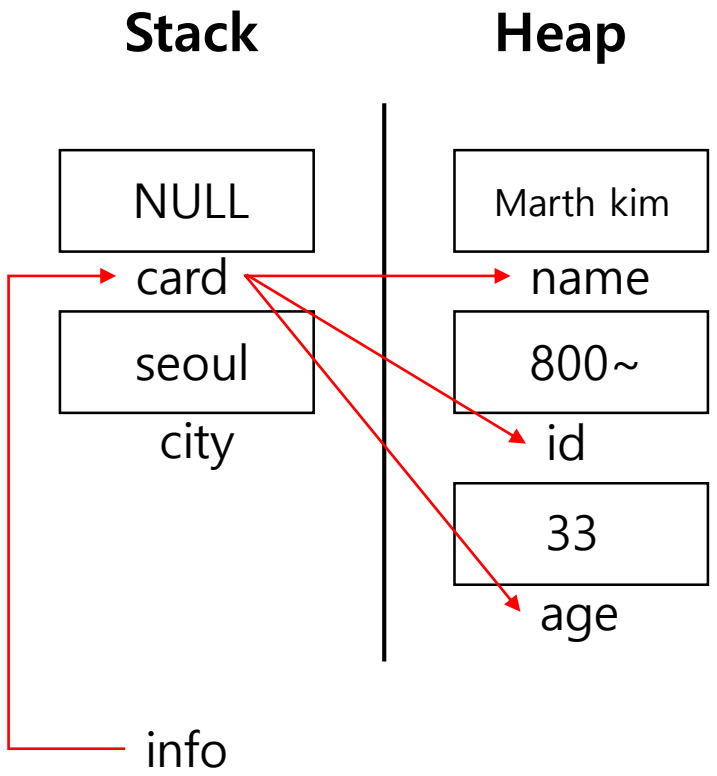
  값이 할당될 때 메모리에 잡힌다. 선언만 해서는 잡히지 않음

```
typedef struct __id_card{
        char name[NAME_LEN];
        char id[ID_LEN];
        unsigned int age;
}id_card;
```

```
mhn@mhn-900X3L: ~/my_proj/c/6_s
#include <stdio.h>
#include <stdlib.h>

typedef struct __id_card{

        char *name;
        char *id;
        unsigned int age;

} id_card;

typedef struct __city{

        id_card *card;
        char city[30];

} city;

int main(void){

        int i;
        city info = {NULL, "Seoul"};
        info.card = (id_card *)malloc(sizeof(id_card));

        info.card->name = "Marth Kim";
        info.card->id = "800903-1012589";
        info.card->age = 33;

        printf("city = %s, name = %s, id = %s, age = %d\n",
                        info.city,info.card->name,info.card->id,info.card->age);

        free(info.card);
        return 0;

}
~
```

```
mhn@mhn-900X3L:~/my_proj/c/6_s$ ./a.out
city = Seoul, name = Marth Kim, id = 800903-1012589, age = 33
mhn@mhn-900X3L:~/my_proj/c/6_s$
```

포인터가 구조체에 접근할 때 ->를 쓴다



7일차 내용 정리

- 함수포인터

  함수의 주소값을 저장할 수 있는 포인터 변수

  쓰는 이유 중 가장 핵심적인 것은 비동기 처리이다.

  비동기 처리는 언제 어떤 이벤트가 발생할지 알 수 없는 것들을 다룬다.

  컴퓨터 관점에서 임기응변을 잘 하도록 만들어주는 것이 바로 함수 포인터다.

  (결론: 비동기 처리 - 함수 포인터)

  다음으로는 HW 개발 관점에서 인터럽트와 시스템 콜(유일한 SW 인터럽트임)이 있다.

  이 두개는 사실상 비동기 동작에 해당된다.

test1.c (~/my_proj/c/7_s) - gedit

Open

```c
#include <stdio.h>

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

int number(void){
        printf("number called\n");
        return 7;
}

void(* bbb(void))(void){
        printf("bbb called\n");
        return aaa;
}

void ccc(void(*p)(void)){
        printf("ccc : I can call aaa!\n");
        p();
}

int (* ddd(void))(void){
        printf("ddd : Ican call number\n");
        return number;
}

int main(void){
        int res;
        bbb()();
        ccc(aaa);
        ddd()();
        res=ddd()();
        printf("res = %d\n",res);
        return 0;
}
```

함수 프로토 타입이란?
리턴, 함수명, 인자에 대한 기술서

void (* bbb(void))(void) ->
void (*)(void) bbb(void)
리턴:void (*)(void)
이름:bbb
인자:void

void ccc(void(*p)(void))
리턴:void
이름:ccc
인자:void(*p)(void)

int (* ddd(void))(void) ->
int (*)(void) ddd(void)
리턴:int (*)(void)
이름:ddd
인자:void

- memmove()

  Memory Move의 합성어

  메모리의 값을 복사할 때 사용함

  memove(목적지, 원본, 길이);

  memcpy보다 느리지만 안정적임

- memcpy()

  memory 공간에 겹치는 부분이 없을 때 사용

  겹치는 부분이 없다면 성능에 좋음

- strlen()

  주로 strcpy()등의 함수와 함께 사용

  문자열의 길이를 구하는데 사용함

  strlen("문자열");

- strcpy(), strncpy()

  문자열을 복사하고 싶을 경우 사용

  strcpy(dst, src), strncpy(dst, src, length)

  *strncpy는 길이를 정해주기 때문에 해킹 위험이 없다.

- strcmp(), strncmp()

  문자열이 서로 같은지 비교하고 싶을때 서로 같은 경우 0을 반환하게됨

  strcmp(str1, str2), strncmp(str1, str2, len)

  *위와 마찬가지로 해킹 위험이 없는 strncmp를 사용하는 것이 좋다.

문제풀기

float (* (* test(void (*p)(void)))(float (*)(int, int)))(int, int)
위와 같은 프로토타입의 함수가 구동되도록 프로그래밍 하시오.

mhn@mhn-900X3L: ~/my_proj/c/7_h

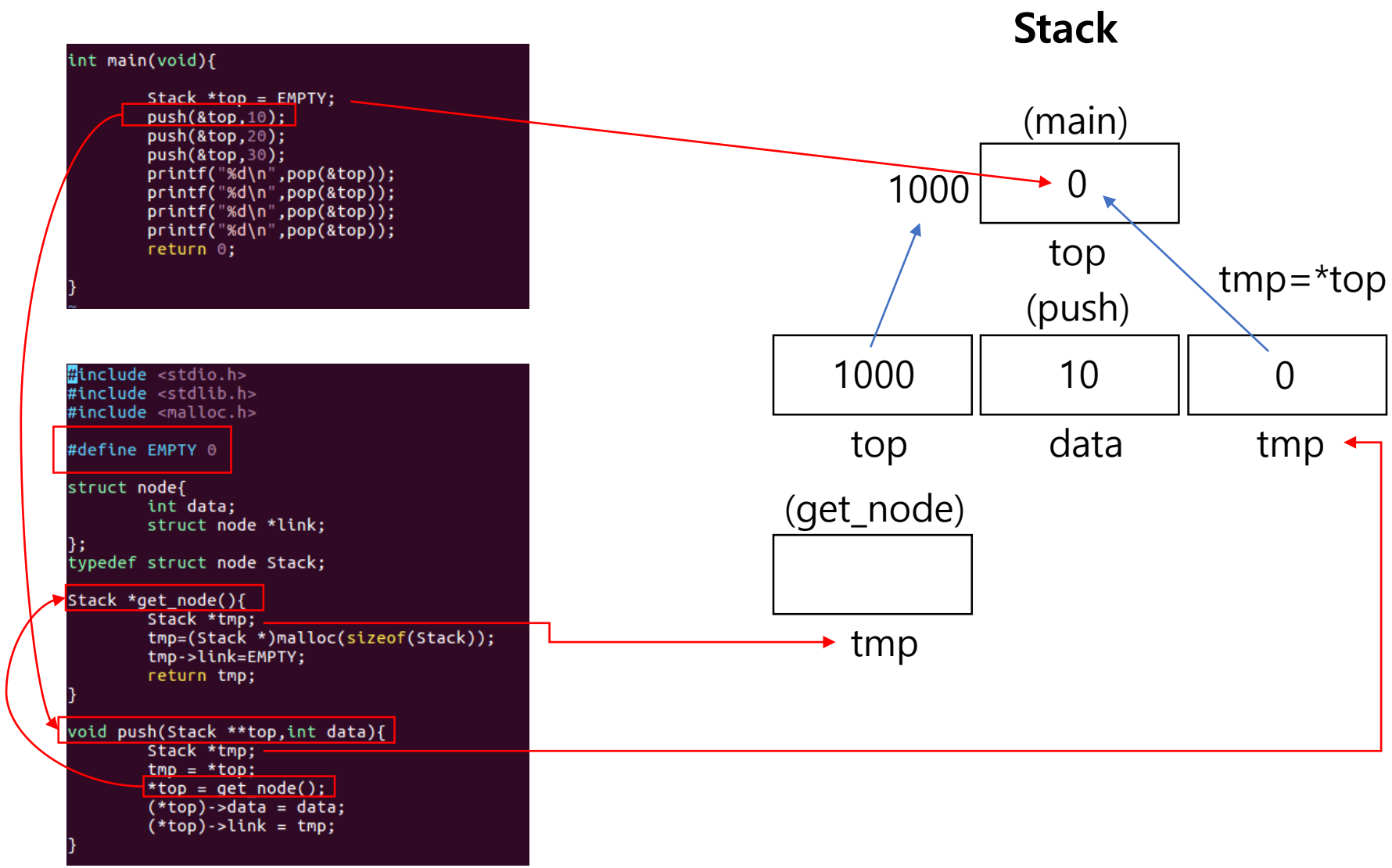
```c
#include <stdio.h>


float test1(int n1,int n2){
return (n1+n2) + 0.2357;
}

float (*test2(float *test)(int,int)){

}

void test3(void){
printf("test3 called\n");
}


//float (*)(int,int)(*)(float (*)(int,int))test(void(*p)(void))
float (*(* test_main(void (*p)(void)))(float (*)(int,int)))(int,int){
p();
return test2;
}

int main(){

float result = test_main(test1)(tes2)(4,5);
printf("result = %f\n",result);

return 0;

}
~
~
```

함수 구현을 못했습니다.

연결리스트 예제 그림 그리기

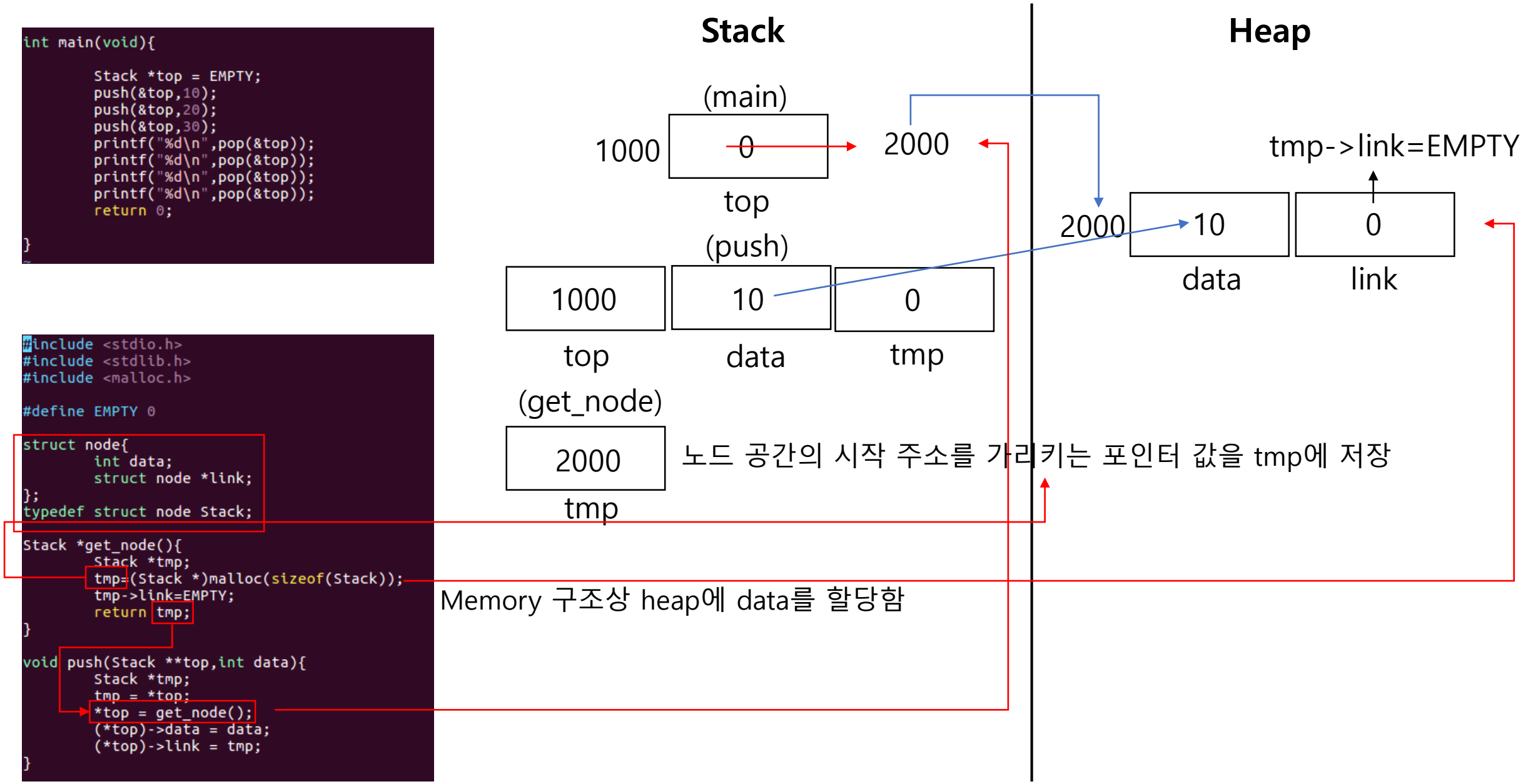
```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

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

#define EMPTY 0

struct node{
        int data;
        struct node *link;
};
typedef struct node Stack;

Stack *get_node(){
        Stack *tmp;
        tmp=(Stack *)malloc(sizeof(Stack));
        tmp->link=EMPTY;
        return tmp;
}

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



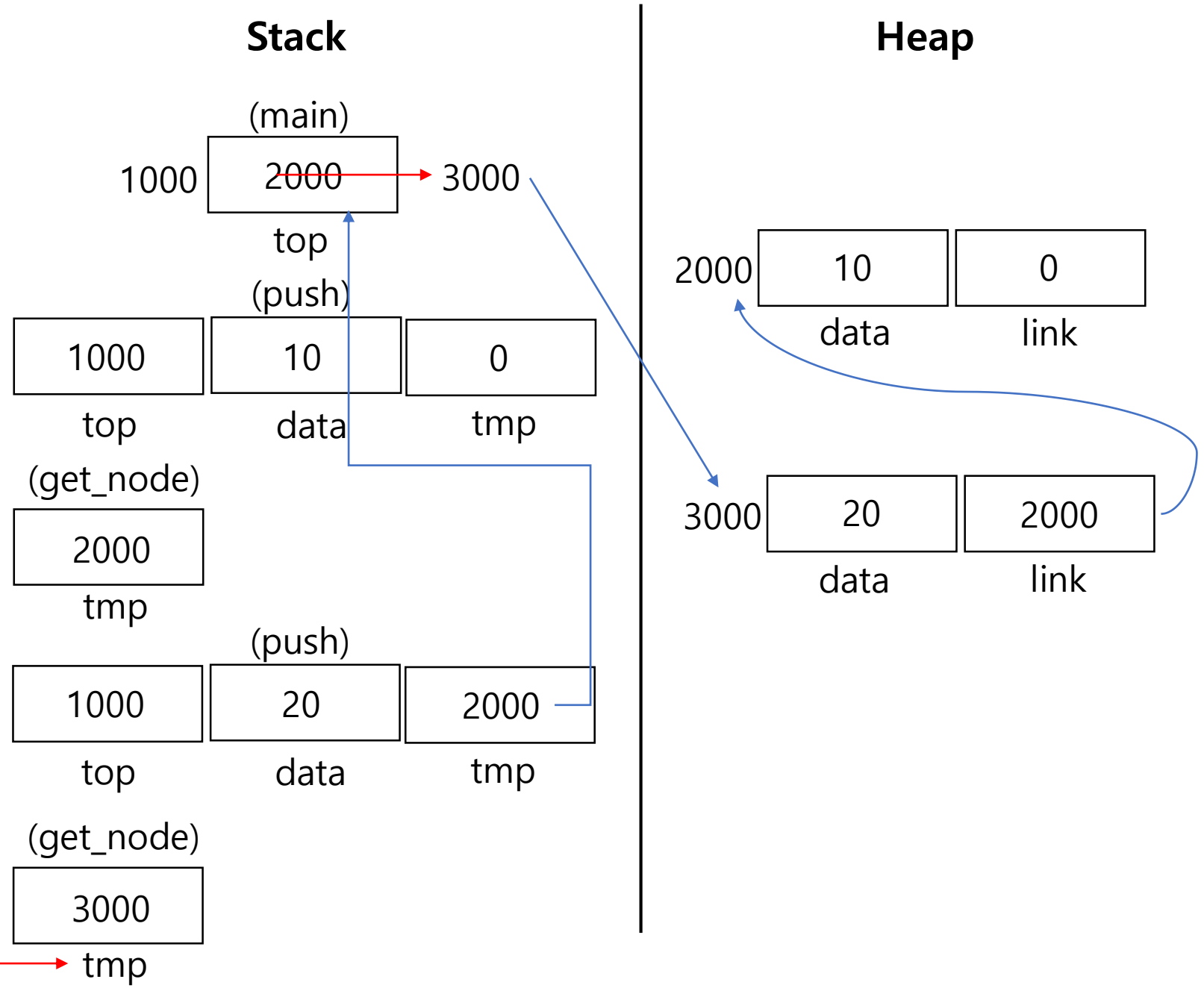
```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

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

#define EMPTY 0

struct node{
        int data;
        struct node *link;
};
typedef struct node Stack;

Stack *get_node(){
        Stack *tmp;
        tmp=(Stack *)malloc(sizeof(Stack));
        tmp->link=EMPTY;
        return tmp;
}

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



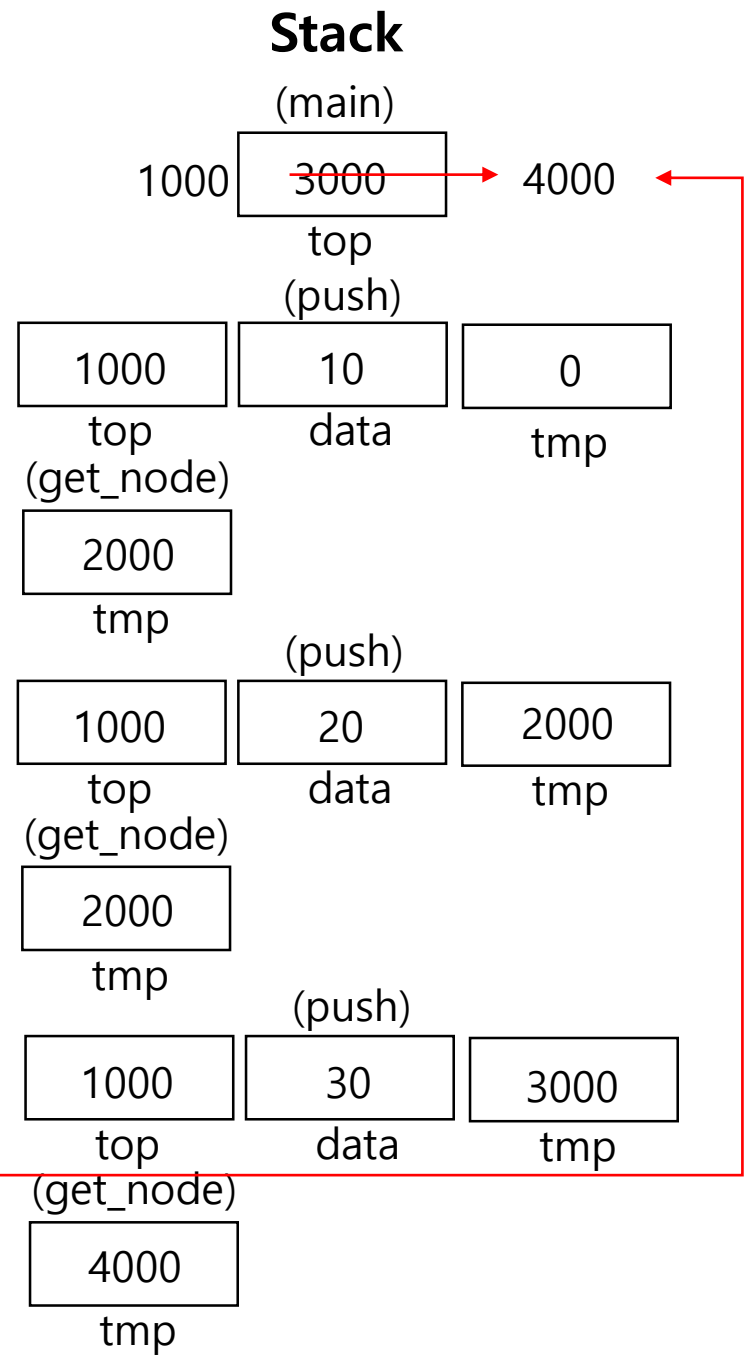
```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

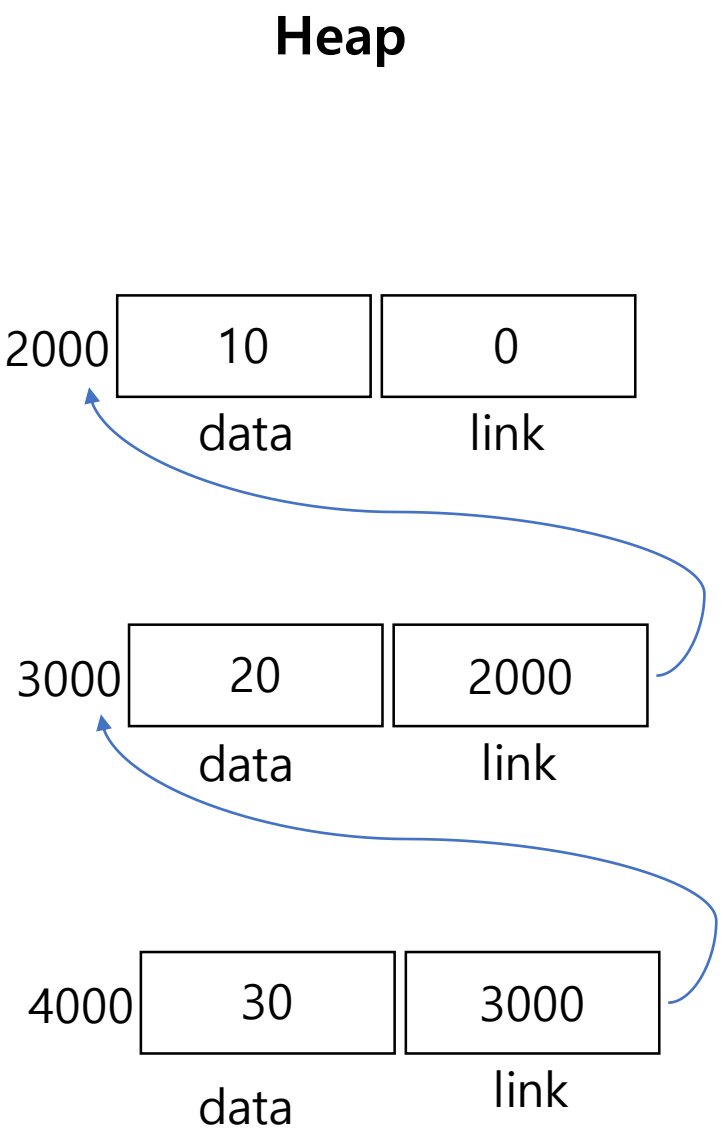
```c
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

#define EMPTY 0

struct node{
        int data;
        struct node *link;
};
typedef struct node Stack;

Stack *get_node(){
        Stack *tmp;
        tmp=(Stack *)malloc(sizeof(Stack));
        tmp->link=EMPTY;
        return tmp;
}

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



```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

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

#define EMPTY 0

struct node{
        int data;
        struct node *link;
};
typedef struct node Stack;

Stack *get_node(){
        Stack *tmp;
        tmp=(Stack *)malloc(sizeof(Stack));
        tmp->link=EMPTY;
        return tmp;
}

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

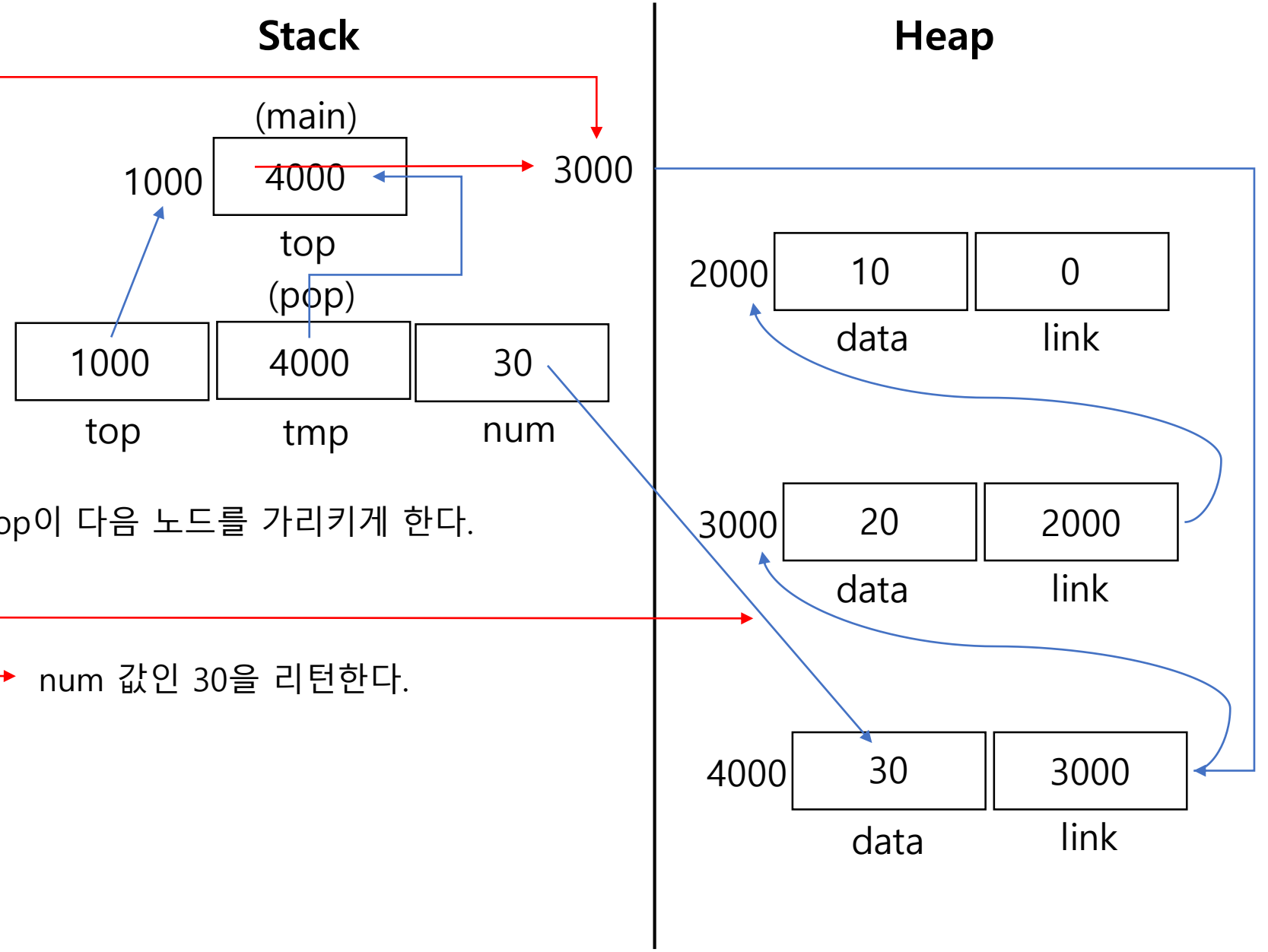




```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

```c
int pop(Stack **top){
        Stack *tmp;
        int num;
        tmp=*top;
        if(*top == EMPTY){
                printf("Stack is empty!!!\n");
                return 0;
        }
        num = tmp->data;
        *top = (*top)->link;
        free(tmp);
        return num;
}
```

```
mhn@mhn-900X3L:~/my_proj/c/7_h$ ./a.out
30
20
10
Stack is empty!!!
0
mhn@mhn-900X3L:~/my_proj/c/7_h$
```

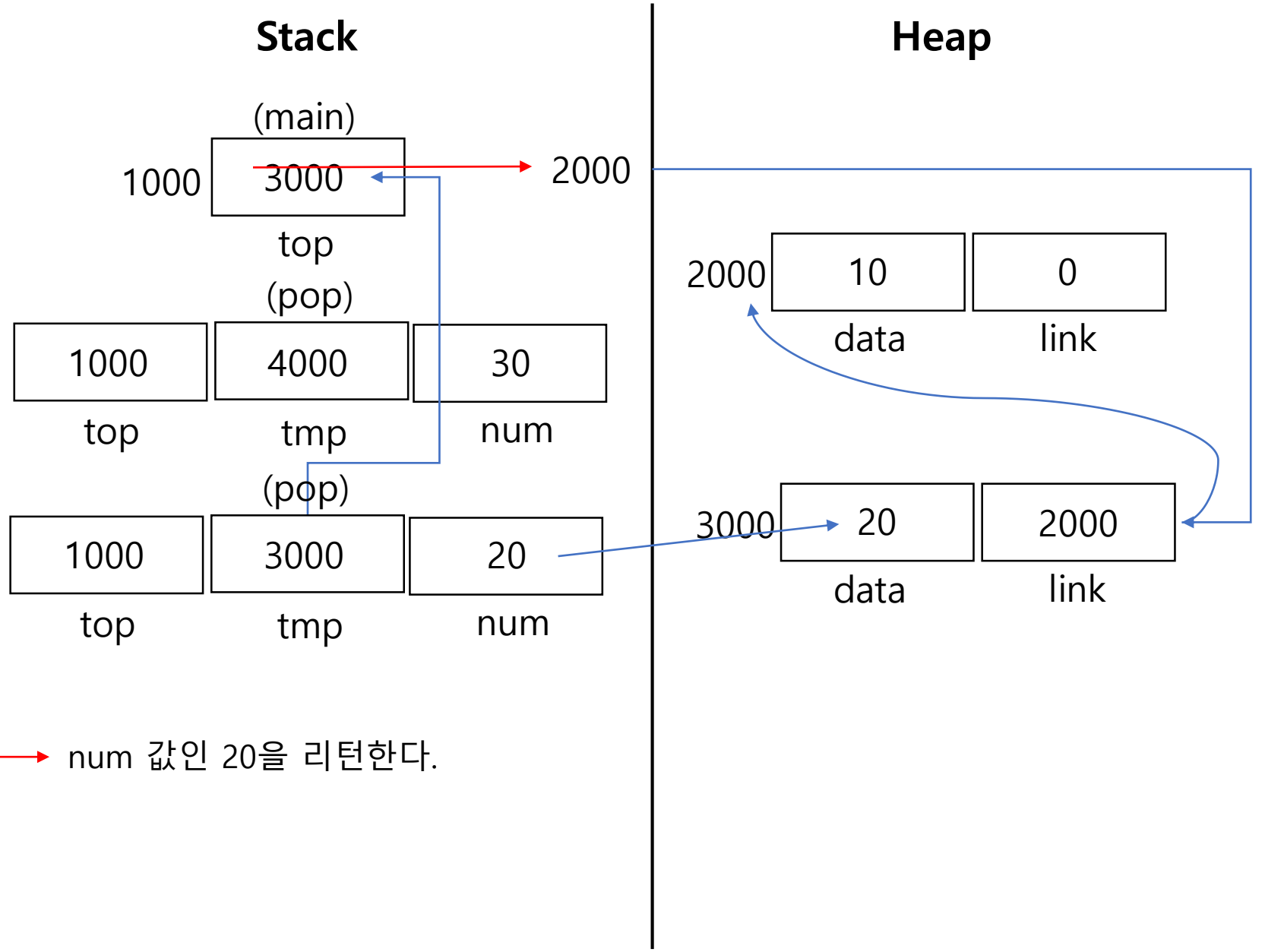


```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

```c
int pop(Stack **top){
        Stack *tmp;
        int num;
        tmp=*top;
        if(*top == EMPTY){
                printf("Stack is empty!!!\n");
                return 0;
        }
        num = tmp->data;
        *top = (*top)->link;
        free(tmp);
        return num;
}
```

```
mhn@mhn-900X3L:~/my_proj/c/7_h$ ./a.out
30
20
10
Stack is empty!!!
0
mhn@mhn-900X3L:~/my_proj/c/7_h$
```

num 값인 20을 리턴한다.

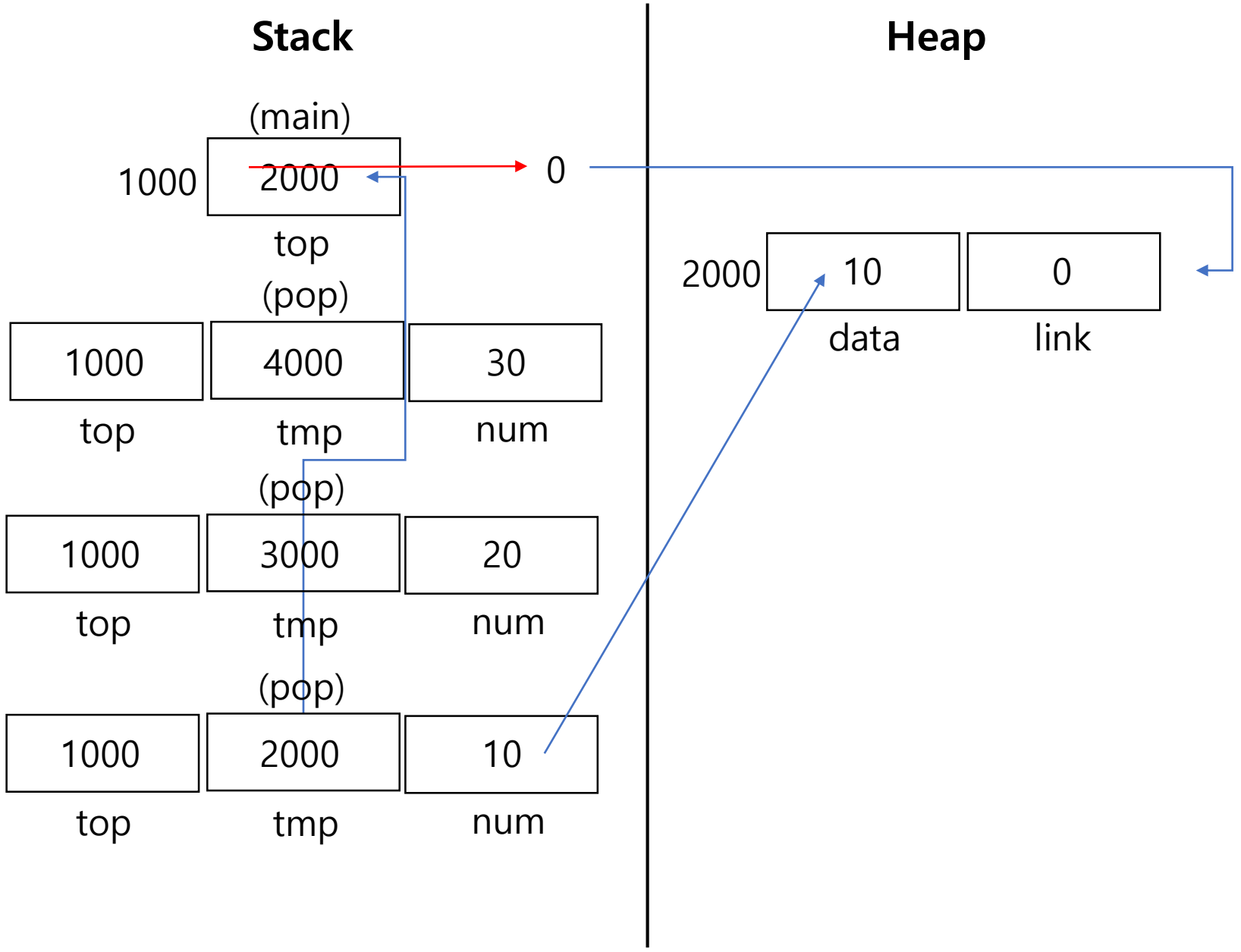


```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

```c
int pop(Stack **top){
        Stack *tmp;
        int num;
        tmp=*top;
        if(*top == EMPTY){
                printf("Stack is empty!!!\n");
                return 0;
        }
        num = tmp->data;
        *top = (*top)->link;
        free(tmp);
        return num;
}
```

num 값인 10을 리턴한다.

```
mhn@mhn-900X3L:~/my_proj/c/7_h$ ./a.out
30
20
10
Stack is empty!!!
0
mhn@mhn-900X3L:~/my_proj/c/7_h$
```



Stack

```c
int main(void){

        Stack *top = EMPTY;
        push(&top,10);
        push(&top,20);
        push(&top,30);
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        printf("%d\n",pop(&top));
        return 0;

}
```

```c
int pop(Stack **top){
        Stack *tmp;
        int num;
        tmp=*top;
        if(*top == EMPTY){
                printf("Stack is empty!!!\n");
                return 0;
        }
        num = tmp->data;
        *top = (*top)->link;
        free(tmp);
        return num;
}
```

```
mhn@mhn-900X3L:~/my_proj/c/7_h$ ./a.out
30
20
10
Stack is empty!!!
0
mhn@mhn-900X3L:~/my_proj/c/7_h$
```

(main)

1000 | 0

top

조건문에 걸린다

(pop)

1000	4000	30
top	tmp	num

(pop)

1000	3000	20
top	tmp	num

(pop)

1000	2000	10
top	tmp	num

(pop)

1000	0	0
top	tmp	num