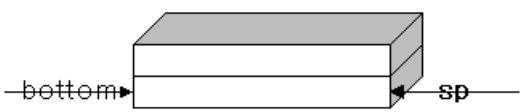
9. scanf()에 &가 있어야 하는가?

- 자료구조data structure는 데이터를 어떻게 표현하고 보관할 것인가를 표현합니다.
- 자료구조 중에서도 스택stack은 빈번하게 이용되므로 그 개념을 파악하는 것은 중요합니다.

스택(stack): 자료구조의 정상

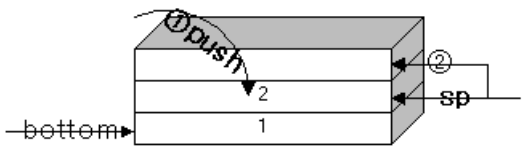
• 빈 스택의 상태는 Stack Bottom Pointer와 SP가 같은 곳을 가리킵니다. 이 때를 스택 비어있음(Stack Empty)이라고 합니다. 이때의 상태를 아래 그림에 나타내었습니다.



스택의 초기 상태(Stack Empty)

- 푸시는 SP가 가리키는 곳에 데이터를 넣고, SP를 1증가시킵니다.

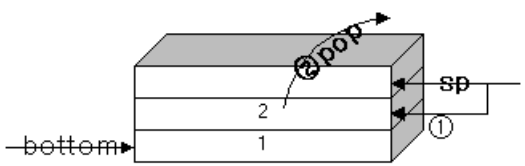
Stack[SP++]=data;



푸시(Push): sp가 가리키는 곳에 데이터를 집어 넣고, sp를 증가시킵니다.

- 팝은 SP를 먼저 감소시키고, SP가 가리키는 곳의 데이터를 꺼냅니다.

return Stack[--SP];



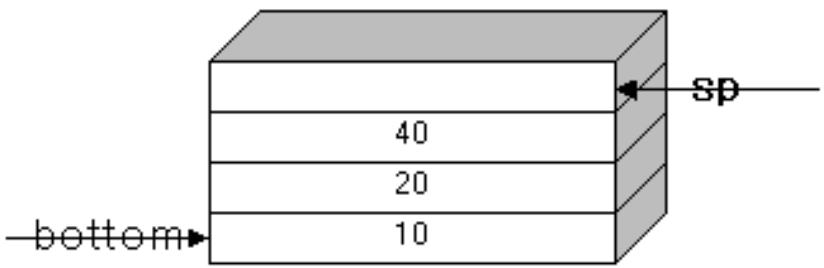
팝(Pop): sp를 먼저 감소시킨 후, sp가 가리키는 곳의 데이터를 리턴합니다.

- 스택은 아래와 같이 구현할 수 있습니다.

```
class CStack {
    int sp;
    //데이터를 관리하기 위해, 자료형의 배열이나 링크드 리스트의 포인터
    //가 선언되어야 한다.
  public:
    CStack() {
        sp=0;
        //일반적으로 동적으로 메모리를 할당하는 문장이 명시된다.
    }
    ~CStack() {
        //동적으로 할당된 메모리를 해제하는 문장이 명시된다.
    }
    void Push(type data);
    type Pop();
};
```

• 정수 데이터에 대한 스택에 관한 아래의 일련의 명령어들의 결과를 그림으로 나타내보면 아래 그림과 같습니다.

```
stack->Push(10);
stack->Push(20);
stack->Push(30);
i=stack->Pop();//30이 팝된다.
stack->Push(40);
stack->Push(50);
i=stack->Pop();//50이 팝된다.
```

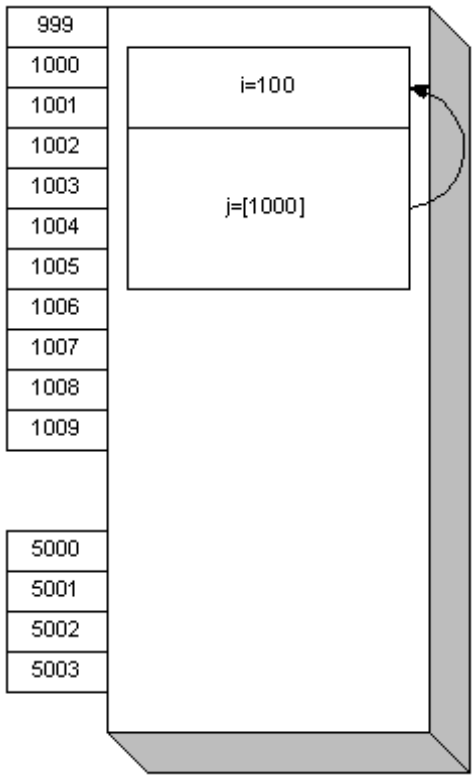


&의 역할

• &는 주소 연산자(address-of operator), C++의 참조 연산자(reference operator) 그리고 비트 곱(bitwise AND) 연산자로 사용됩니다.
• 주소연산자 &는 피연산자(operand)의 주소를 계산합니다.

```
#include <stdio.h>

void main() {
    short i,*j;

    j=&i;//i의 주소가 j에 할당된다.
    *j=100;
    printf("%d\n",i);//100
}//main
```

- 위 프로그램의 메모리 구조는 아래와 같습니다.



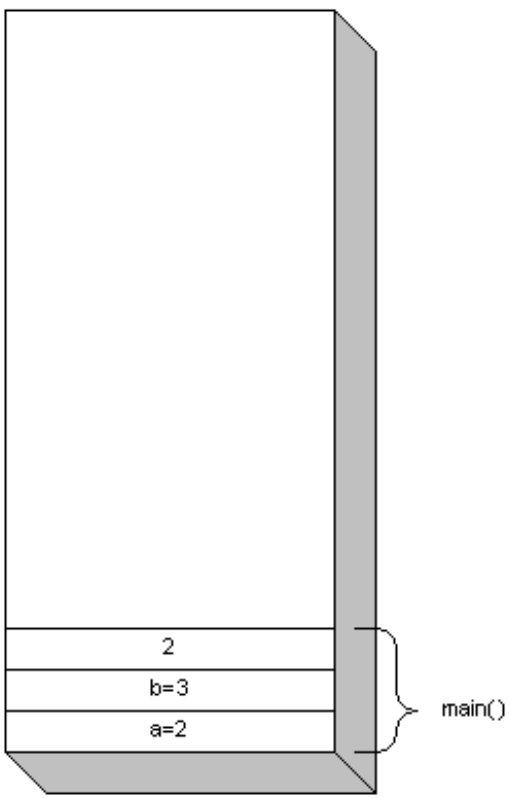
메모리 상태: i는 2바이트를 차지하지만, j는 포인터이므로 4바이트를 차지합니다.

- 두 개의 변수의 값을 바꾸는(swap) 함수 Swap()을 만들어 보기로 합시다.
- 이 함수는 두 개의 정수 a,b를 받아 a와 b의 값을 교환합니다.

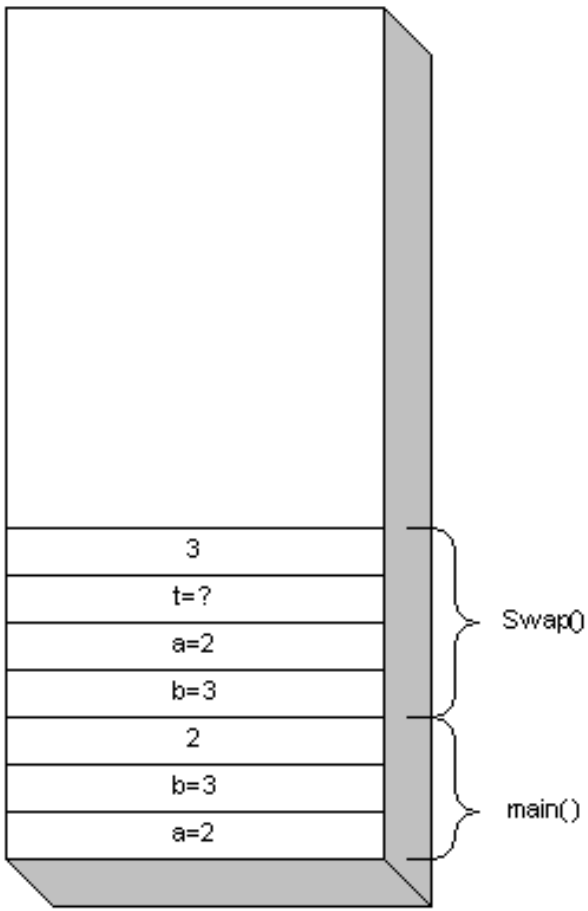
```
#include <stdio.h>

void Swap(short a,short b) {
    short t=a;
    a=b;
    b=t;
}//Swap

void main() {
    short a=2,b=3;
    Swap(a,b);
    printf("%d,%d\n",a,b);
}//main
```

- 결과는 다음과 같습니다.

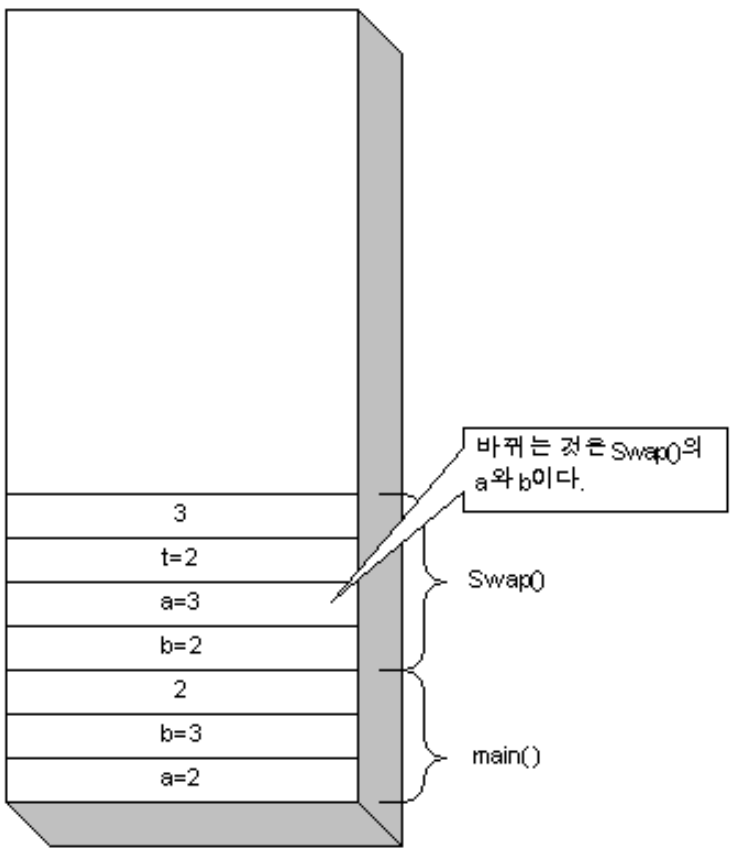
• main() 함수의 블록에 진입했을 때, 블록 안에서 선언된 2개의 지역 변수는 스택에 할당될 것입니다. 그리고 각각 2,3으로 초기화 될 것입니다. 이 상태를 아래 그림에 나타내었습니다.



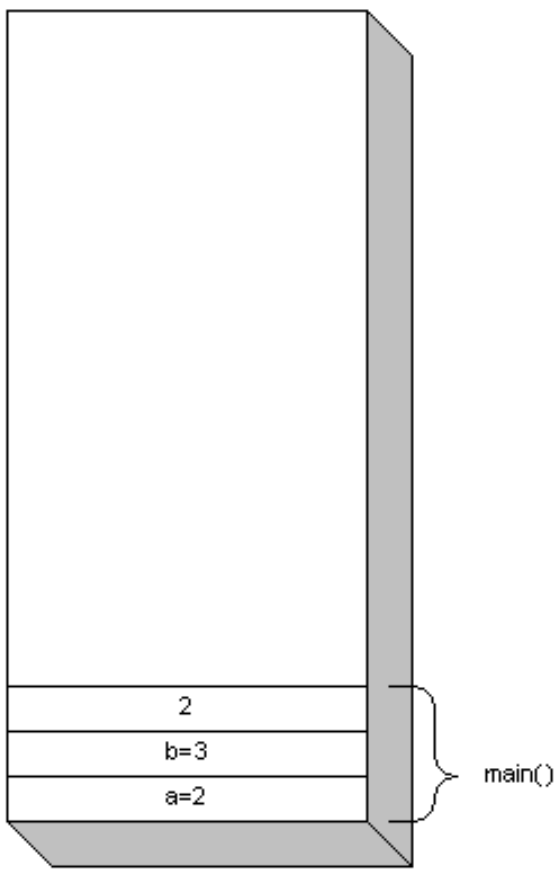
- 스택에는 차례대로 b,a,t가 푸시됩니다. 이 상태를 아래 그림에 나타내었습니다.



• 먼저 a의 값이 t에 할당됩니다. a에는 b의 값이 할당되므로 a==3이 됩니다. b에는 보존해 둔 a값인 t가 할당되므로, b==2가 됩니다.



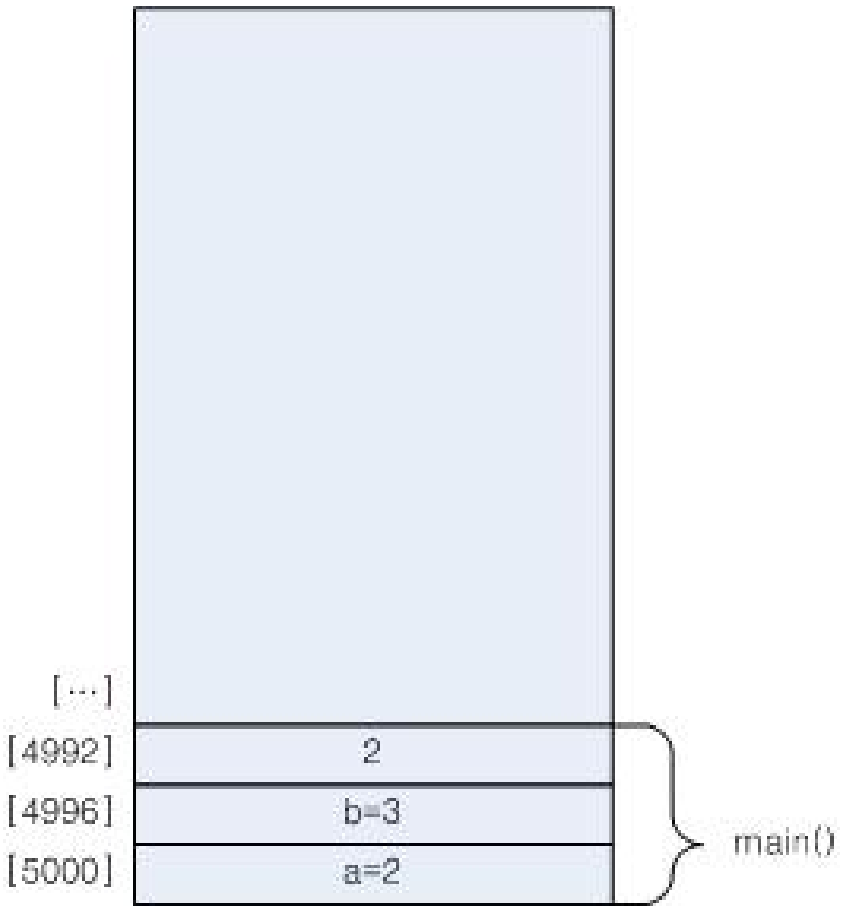
• 이제 작업을 끝내고 Swap() 함수를 빠져 나옵니다. 스택에 상단에 3이 있으므로, 이것을 팝pop한 후에 다시 3개를 스택에서 팝pop할 것입니다.



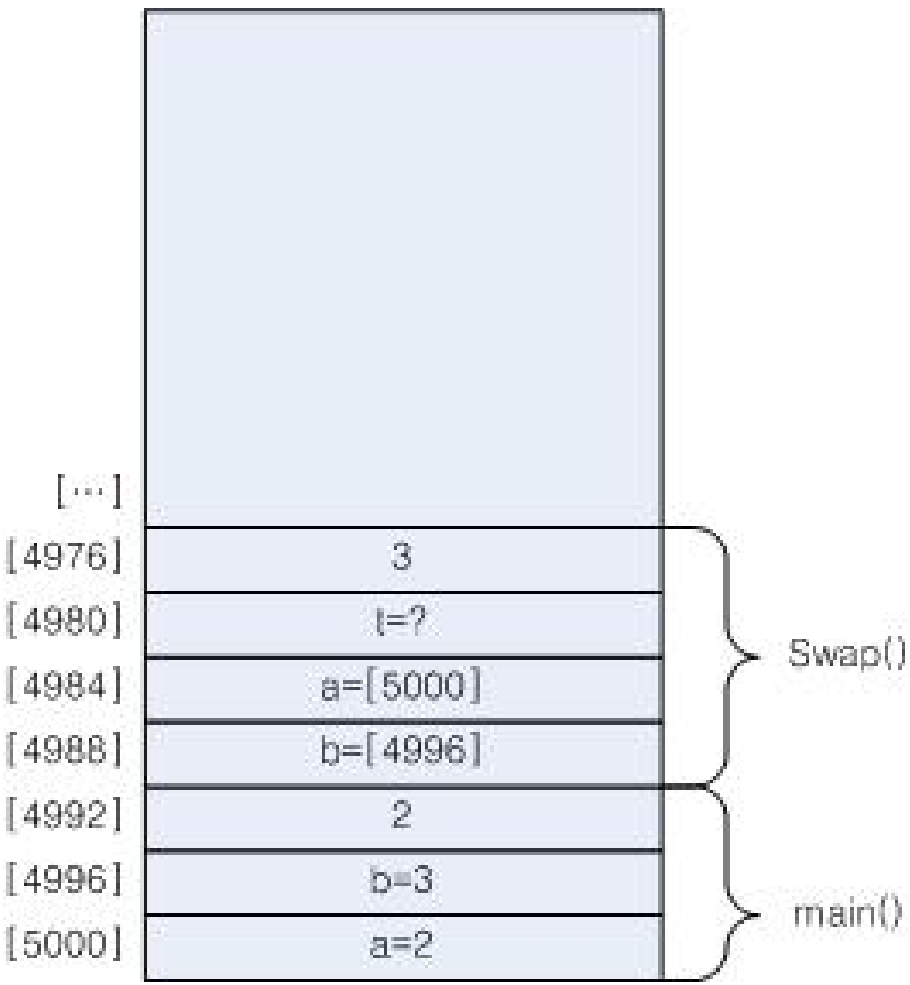
• main()의 지역 변수인 a와 b를 바꾸기 위해서는 함수로 무엇을 전달해야 할까요? 그렇습니다.
• a의 주소와 b의 주소를 전달해야 합니다

```
#include <stdio.h>

void Swap(int *a,int *b) {
    int t=*a;//a가 가리키는 값을 t에 대입한다.
    *a=*b;//b가 가리키는 값을 a가 가리키는 곳에 대입한다.
    *b=t;//t를 b가 가리키는 곳에 대입한다.
}//Swap

void main() {
    int a=2,b=3;
    Swap(&a,&b);//a와 b의 주소를 전달해야 한다.
    printf("%d,%d\n",a,b);
}//main
```

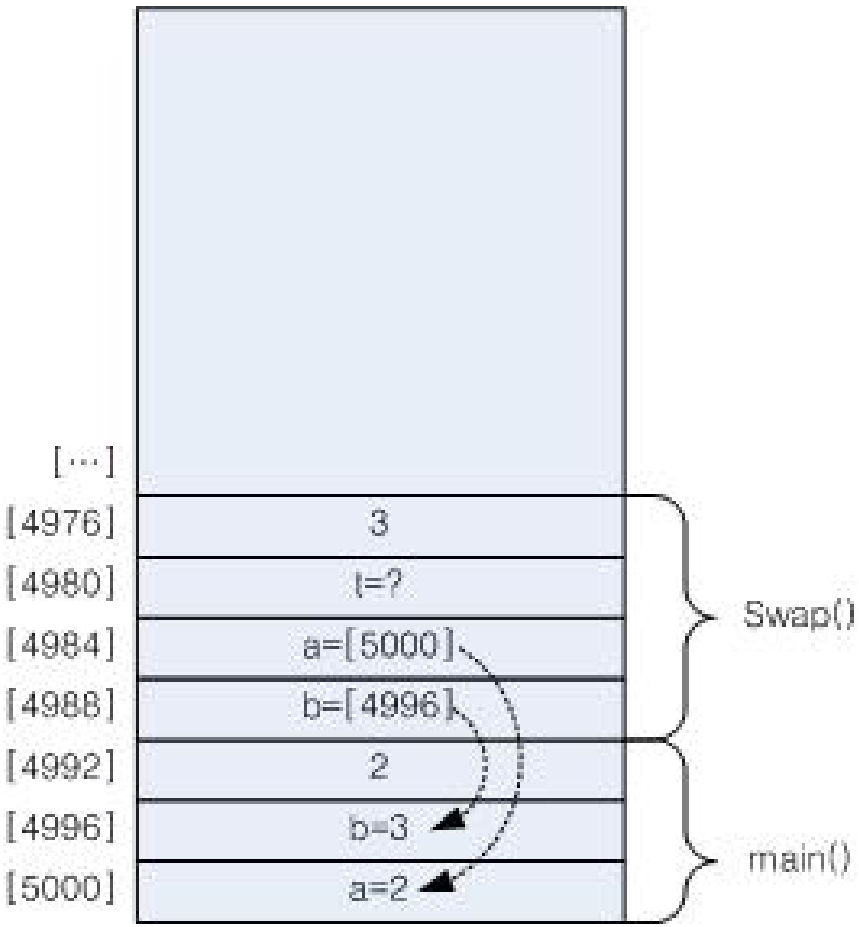
• main() 블록에 진입했을 때, 두 개의 지역 변수는 스택에 푸시push됩니다. a가 스택의 5000번지에 할당된다면, b는 다음에 푸시push되므로, 4996번지에 할당될 것입니다.



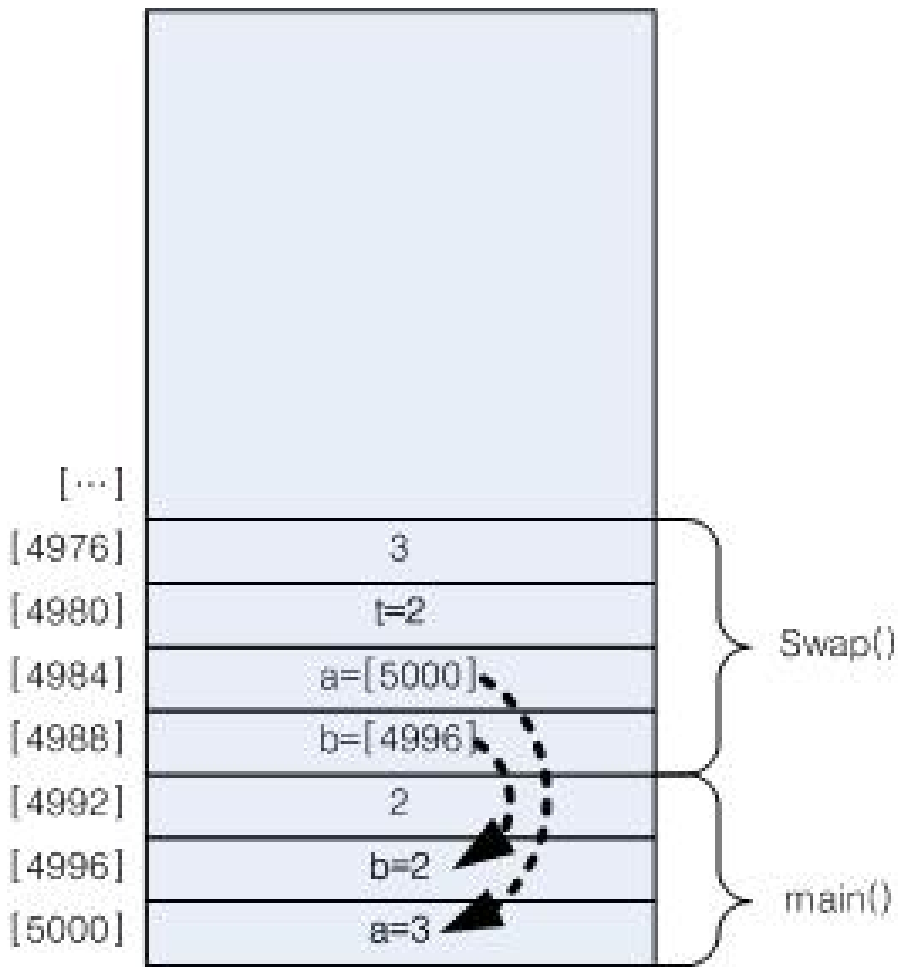
- main()에서 Swap()을 호출하고 있습니다.



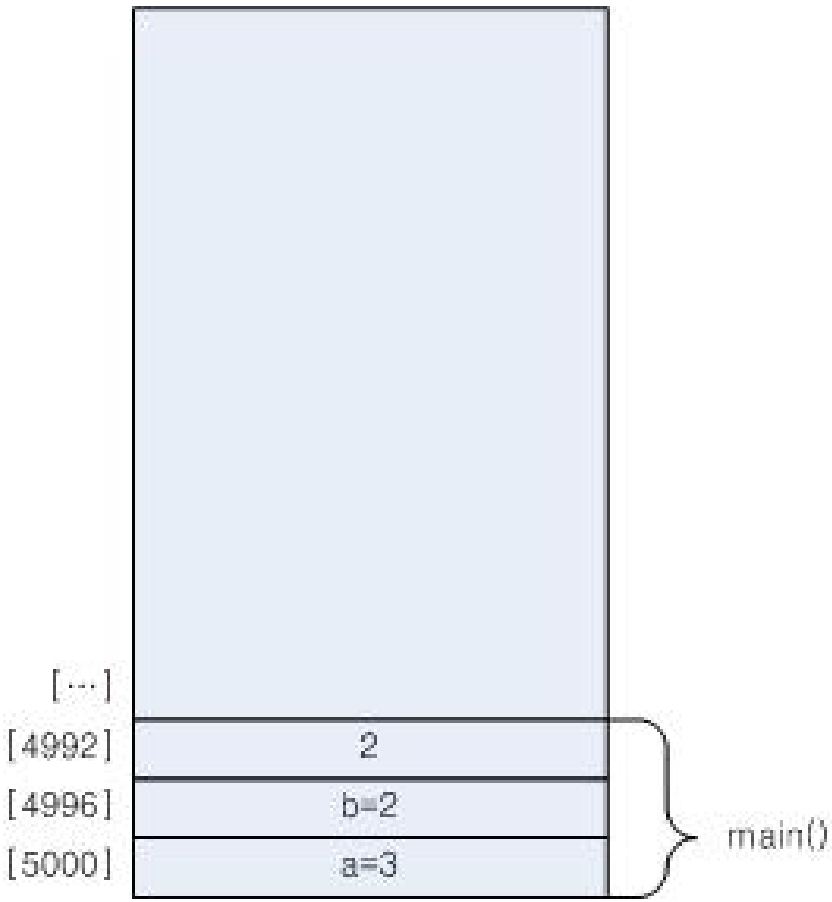
• Swap()의 a는 main()의 a와 다릅니다. Swap()의 a는 main()의 a를 가리키는 포인터입니다.



- main()의 a,b를 교환하기 위해서는 이제 간접 지정 연산자 *를 사용해야 합니다.



• Swap()함수를 종료하는 순간 메모리의 상태는 Swap()을 호출하기 전의 메모리 상태로 회복될 것입니다.



• 함수를 호출할 때, 이렇게 변수의 값을 전달하는 것이 아니라, 변수의 주소를 전달하는 것을 **주소에 의한 호출(call by address)**이라고 불러도 무난합니다. 하지만 이것은 엄격하게 값에 의한 호출입니다. 주소값을 전달한 것이니까요.
• 이것이 **참조에 의한 호출(call by reference)**이 아니라는 것에 주의하세요. 변수의 참조가 넘어간 것은 아닙니다.
• 함수의 파라미터로 넘어간 변수를 함수 안에서 바꾸었을 때, 함수를 빠져 나왔을 때에도 값이 바뀌어지도록 하려면, 변수의 주소를 전달해야 합니다.

scanf()에 &를 적어야 하는 이유

• 키보드를 통해 정수를 한 개 입력받는 함수를 아래와 같이 만들 수 있습니다.

```
int i;
scanf("%d",i);
```

• 무엇이 잘못되었을까요? 사용자가 입력한 변수는 scanf() 내부에서 분명히 i에 할당될 것입니다. 즉 값이 변경될 것입니다. 하지만, scanf()를 빠져 나오는 순간 i의 값은 스택에서 팝pop되므로, scanf()를 호출하기 전의 i값으로 남아 있을 것입니다. scanf()에서 바뀐 값이 scanf()를 빠져 나왔을 때에도 바뀌게 하려면, i의 주소, &i를 전달해야 합니다.

```
int i;
scanf("%d",&i);
```

진보된 주제: 포인터를 바꾸려면?

• 바꾸어야 하는 값이 정수의 포인터라면, 정수의 포인터를 가리키는 포인터(int **)를 전달해야 합니다.

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

void main() {
    int *ip;

    ip=(int *)malloc(sizeof(int));
      //ip는 int *이므로 (int *)를 사용해서 명시적인 형변환이 반드시 필
      //요하다.
    *ip=100;
    printf("%d\n",*ip);
    free(ip);
}//main
```

new를 사용한 소스

```
#include <stdio.h>

void main() {
    int *ip;

    ip=new int;//int만큼, 즉 4바이트를 할당해서 시작 주소를 ip에 대입한
               //다.
    *ip=100;
    printf("%d\n",*ip);
    delete ip;
}//main
```

• 이제 메모리 할당하는 부분을 하나의 함수로 만들기로 했습니다. 그래서 소스를 아래와 같이 수정하였습니다.

```
#include <stdio.h>

void f(int *ip) {
    ip=new int;
}//f

void main() {
    int *ip;

    f(ip);
    *ip=100;
    printf("%d\n",*ip);
    delete ip;
}//main
```

- f()에서 바꾸어야 하는 값은 ip가 가리키는 값(int)이 아니라, ip(int *)입니다.
- 정수 포인터(int *)를 바꾸어야 하므로, 함수 f()는 잘못 작성된 것입니다.
- f()는 정수 포인터**의 포인터**를 전달받아야 하며, main()에서 f()를 호출할 때 파라미터에는 ip를 전달해서는 안되고, ip의 주소인 &ip를 전달해야 합니다.
- &ip는 정수 포인터의 포인터이므로 f()쪽에서 다음과 같이 선언해야 합니다.

int **

```
#include <stdio.h>

void f(int **ip) {
    *ip=new int;
}//f

void main() {
    int *ip;

    f(&ip);//ip의 주소를 넘겨 주어야 한다.
    *ip=100;
    printf("%d\n",*ip);
    delete ip;
}//main
```

실습문제

1. 다음 프로그램의 잘못된 부분을 수정하세요.(Hint: 포인터 변수는 사용하기 전에 할당(allocation)해야 합니다)

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

void main() {
    int *i;
    scanf("%d",i);
}
```

---

2. 아래 프로그램의 잘못된 부분을 수정하세요.

```
#include <stdio.h>

void f(int **ip) {
    *ip=new int;
}//f

void main() {
    int **ip;

    f(&ip);
    **ip=100;
    printf("%d\n",**ip);
    delete ip;
}//main
```

@