

-Part3-

제4장 동적 메모리 할당과 가변 인 자

학습목차

4.1 동적 메모리 할당

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자

4.1 동적 메모리 할당

4.1 동적 메모리 할당

- 배울 내용

① 프로세스의 메모리 공간

② 동적 메모리 할당의 필요성

4.1 동적 메모리 할당 (1/6)

• 프로세스의 메모리 구조

코드 영역
(실행 코드, 함수)

스택 영역
(지역 변수, 매개 변수)

데이터 영역
(전역 변수, 정적 변수)

힙 영역
(동적 메모리 할당)

- **코드 영역:** 프로그램 실행 코드, 함수들이 저장되는 영역
- **스택 영역:** 매개변수, 지역 변수, 중괄호(블록) 내부에 정의된 변수들이 저장되는 영역
- **데이터 영역:** 전역 변수, 정적 변수들이 저장되는 영역
- **힙 영역:** 프로그램이 실행 되는 동안 동적으로 메모리를 할당할 수 있는 영역

4.1 동적 메모리 할당 (2/6)---[4-1.c 실습]

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

```
int a=10;                // 전역 변수 a 선언
```

```
int main(void)
```

```
{
```

```
    int num1=10, num2=20;    // 지역변수 num1, num2 선언
```

```
    static int s=20;        // 정적 변수 s 선언
```

```
    printf("데이터 출력 : %d %d %d %d \n", a, num1, num2, s);
```

```
    printf("코드 영역 : %x %x \n", main, printf);    // 함수 이름
```

```
    printf("스택 영역 : %x %x \n", &num1, &num2);    // 지역 변수
```

```
    printf("데이터 영역 : %x %x \n", &a, &s);    // 전역 변수, 정적 변수
```

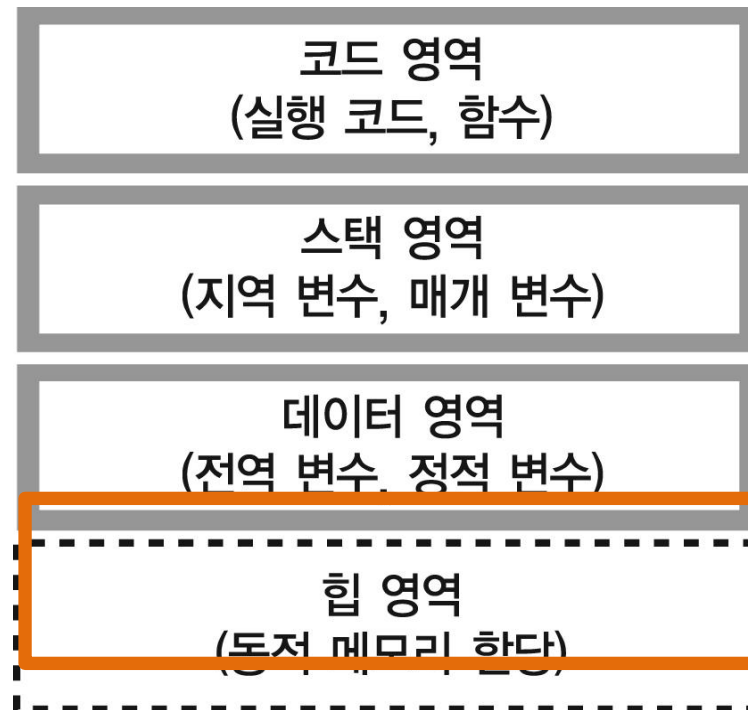
```
    return 0;
```

```
}
```

4.1 동적 메모리 할당 (3/6)

- 동적 메모리 할당

- '힙영역에 할당 된다.'
- '런타임 중(실행 시간)에 이루어 진다.'
- '프로그래머가 동적 메모리 할당을 요구한다.'



4.1 동적 메모리 할당

- 배울 내용

- ① 프로세스의 메모리 공간

- ② 동적 메모리 할당의 필요성

4.1 동적 메모리 할당 (4/6)

- 동적 메모리 할당이 필요한 이유

1. 선언된 배열 요소의 수가 사용된 요소 수 보다 많은 경우
(메모리 낭비)

```
int array[5];           // 선언된 배열 요소 수: 5개(20바이트)  
array[0]=10, array[1]=20, array[2]=30; // 사용된 배열 요소 수: 3개(12바이트)
```

2. 선언된 배열 요소의 수가 사용된 요소의 수보다 적은 경우
(메모리 부족)

```
int array[2];           // 선언된 배열 요소 수: 2개(8바이트)  
array[0]=10, array[1]=20, array[2]=30; // 사용된 배열 요소 수: 3개(12바이트)
```

4.1 동적 메모리 할당 (5/6)

- 동적 메모리 할당이 필요한 이유

3. 배열 선언 시 배열 길이에 변수를 설정한 경우 **에러** 발생

```
int a=5;  
int array[a]    // 배열 선언 시 배열 a를 배열 길이로 사용
```

```
void init(int a)  
{  
    int array[a]; // 배열 선언 시 함수의 인자(지역 변수) a를 배열 길이로 사용
```



결론: '프로그래머가 필요한 메모리 크기를 예측할 수 없다.'
따라서 '동적 메모리 할당의 필요하다.'

4.1 동적 메모리 할당 (6/6)

- 동적 메모리 할당 함수와 해제 함수

– 헤더파일 : **stdlib.h**

종류	함수	성공	실패
메모리 할당 함수	<code>#include <stdlib.h></code> <code>void* malloc (size_t size)</code>	할당된 메모리의 시작 주소 반환	NULL 반환
메모리 할당 함수	<code>#include <stdlib.h></code> <code>void* calloc (size_t num, size_t size)</code>	할당된 메모리의 시작 주소 반환	NULL 반환
메모리 할당 함수	<code>#include <stdlib.h></code> <code>void* realloc (void* p, size_t size)</code>	재할당된 메모리의 시작 주소 반환	NULL 반환
메모리 해제 함수	<code>#include <stdlib.h></code> <code>void free (void* p)</code>	할당된 메모리 해제	-

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자

- 배울 내용

- ① malloc() 함수와 free() 함수
- ② calloc() 함수와 free() 함수
- ③ realloc() 함수와 free() 함수
- ④ 가변 인자

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (1/22)

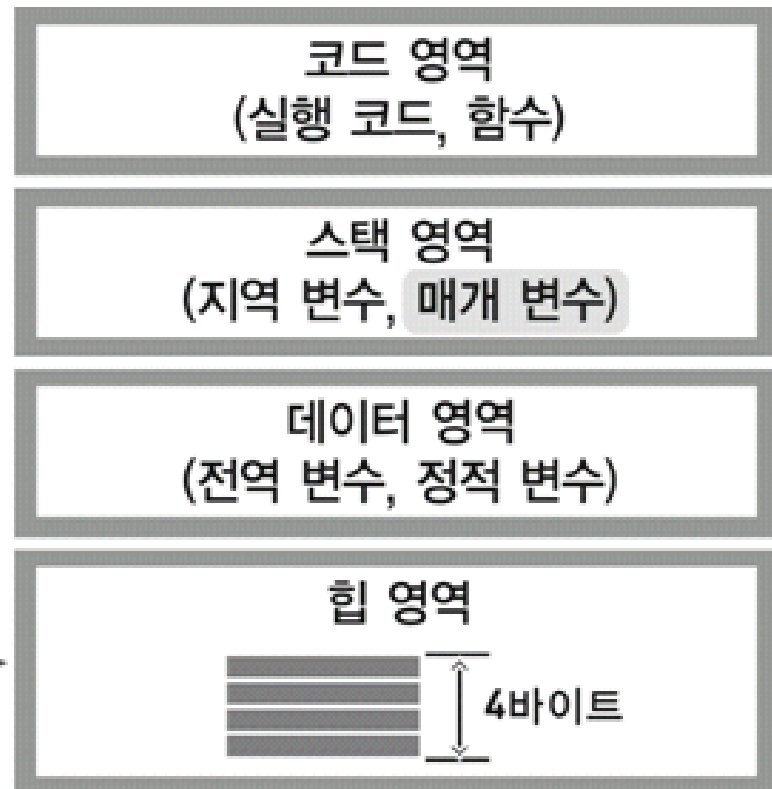

- **malloc()함수와 free()함수**

종류	함수	성공	실패
메모리 할당 함수	void* malloc (size_t size);	할당된 메모리의 시작 주소 반환	NULL 반환
메모리 해제 함수	void free (void* p);	할당된 메모리 해제	

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (2/22)

- **malloc(4) 함수를 이용한 동적 메모리 할당**

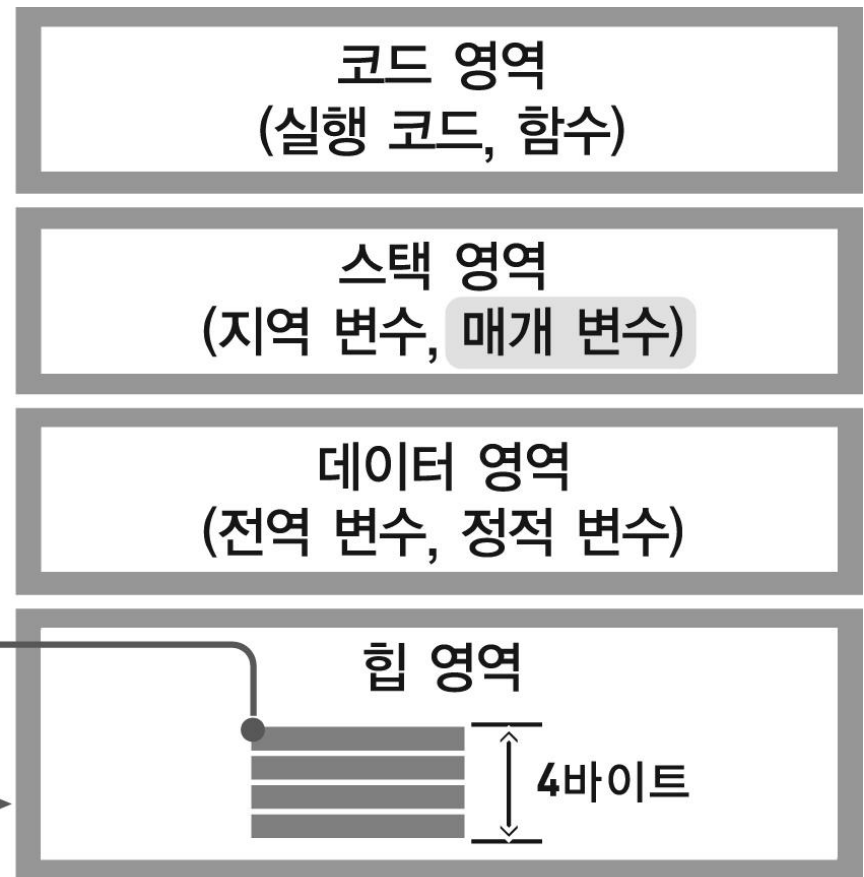
```
int main(void)
{
    int* p=NULL;
    p=(int*) malloc(4);
    return 0;
}
```



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (3/22)

- 할당된 메모리의 시작 주소 형변환 (`int*`)

```
int main(void)
{
    int* p=NULL;
    p=(int*) malloc(4);
    return 0;
}
```



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (4/22)

• free() 함수를 이용한 동적 메모리 해제

```
int main(void)
{
    int* p=NULL;
    p=(int*) malloc(4);
    ...
    free(p);
    return 0;
}
```



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (5/22)

[4-2.c 실습]

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* p=NULL;
    6행 p = (int*)malloc(4);

    if(p==NULL)
        printf("힙 영역에 동적 메모리 할당 실패\n");

    11행 *p = 10;
    printf("주소 : %x \n", p);
    printf("값 : %d \n", *p);

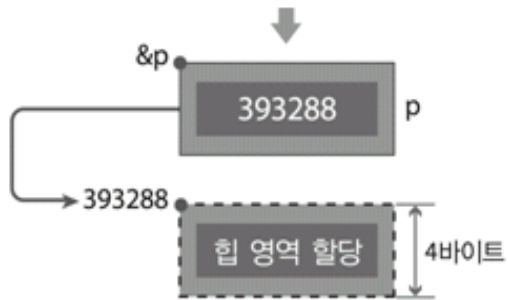
    15행 free(p);
    16행 p = NULL;

    return 0;
}
```

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (6/22)

• 동적 메모리 할당---[4-2.c 분석]

```
p = (int*) malloc(4);
```



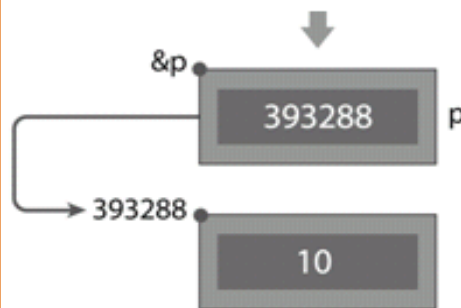
6행 : 힙 영역에 동적 메모리를 할당

```
free(p);  
p=NULL;
```



15행 ~ 16행 : 할당된 동적 메모리를 해제

```
*p = 10;
```



11행 : 할당된 동적 메모리에 데이터 10을 저장

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (7/22)

[4-3.c 실습]

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char* p1 = (char*)malloc(2);
    int* p2 = (int*)malloc(8);

    p1[0] = 'A';    // *(p1+0) = 'A';
    p1[1] = 'B';    // *(p1+1) = 'B';
    p2[0] = 10;     // *(p2+0) = 10;
    p2[1] = 20;     // *(p2+1) = 20;

    printf("주소: %x %x %x %x \n", &p1[0], &p1[1], &p2[0], &p2[1]);
    printf("값: %d %d %d %d \n", p1[0], p1[1], p2[0], p2[1]);

    free(p1);
    p1 = NULL;

    free(p2);
    p2 = NULL;
    return 0;
}
```

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자

- 배울 내용

- ① malloc() 함수와 free() 함수

- ② calloc() 함수와 free() 함수

- ③ realloc() 함수와 free() 함수

- ④ 가변 인자

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (8/22)

• calloc() 함수를 이용한 동적 메모리 할당

종류	함수	반환 값
메모리 할당 함수	<code>void* calloc(size_t num, size_t size);</code>	성공 : 할당된 메모리의 시작 주소 반환 실패 : NULL 반환

– calloc() 함수와 malloc() 함수와의 차이

`int* p1=calloc(4, 4);` → 함수의 입력 인자 2개

==

`int* p2=malloc(16);` → 함수의 입력 인자 1개

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (9/22)

[4-4.c 실습]

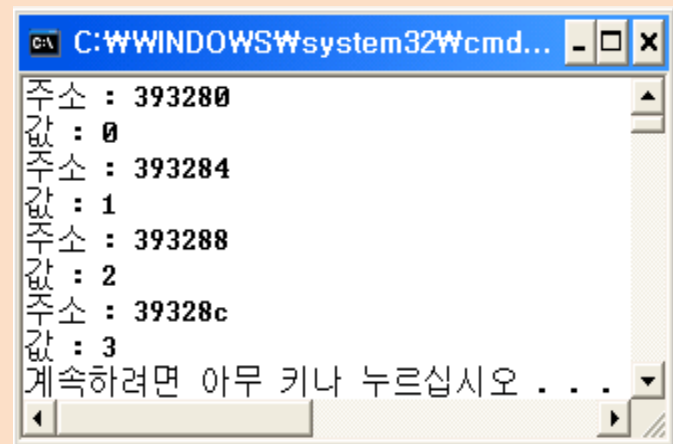
```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i=0;
    int* p=(int*)calloc(sizeof(int), sizeof(int));

    if(p==NULL)
        printf("힙 영역에 동적 메모리 할당 실패\n");

    for(i=0; i<4; i++)
    {
        p[i]=i;      // *(p+i)=i;
        printf("주소: %x \n", &p[i]);
        printf("값: %d \n", p[i]);
    }

    free(p);
    p=NULL;

    return 0;
}
```



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (10/22)

[4-5.c 실습]

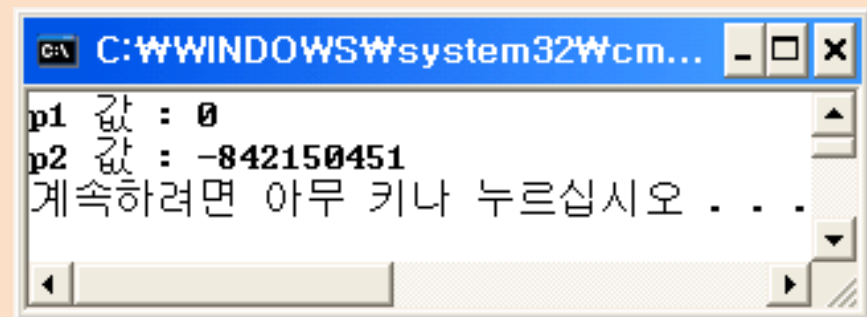
```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* p1=( int*) calloc(1, sizeof(int));
    int* p2=(int*) malloc(4);

    printf("p1 값: %d \n", *p1);
    printf("p2 값: %d \n", *p2);

    free(p1);
    p1=NULL;

    free(p2);
    p2=NULL;

    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
p1 값 : 0
p2 값 : -842150451
계속하려면 아무 키나 누르십시오 . . .
```


4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자

- 배울 내용

- ① malloc() 함수와 free() 함수

- ② calloc() 함수와 free() 함수

- ③ realloc() 함수와 free() 함수

- ④ 가변 인자

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (11/22)

- **realloc()함수를 이용한 동적 메모리 재할당**
 - malloc(), calloc()함수는 동적 메모리를 할당 후 **메모리 변경 불가**
 - **realloc() 함수로 해결**

종류	함수	반환 값
메모리 할당 함수	<code>void* realloc(void* p, size_t size);</code>	성공 : 재할당된 메모리의 시작 주소 반환 실패 : NULL 반환

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (12/22)

[4-6.c 실습]

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
```

```
    int i=0;
```

```
    int* p=(int*) malloc(sizeof(int)*2);
```

```
    p[0]=10;
```

```
    p[1]=20;
```

```
    p=(int*) realloc(p, sizeof(int)*4);
```

```
    p[2]=30;
```

```
    p[3]=40;
```

```
    for(i=0; i<4; i++)
```

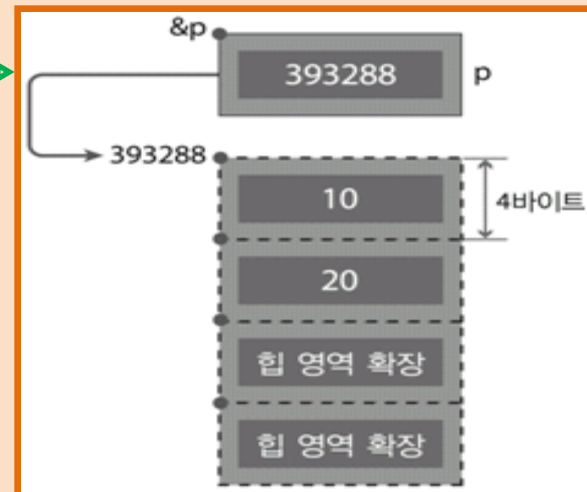
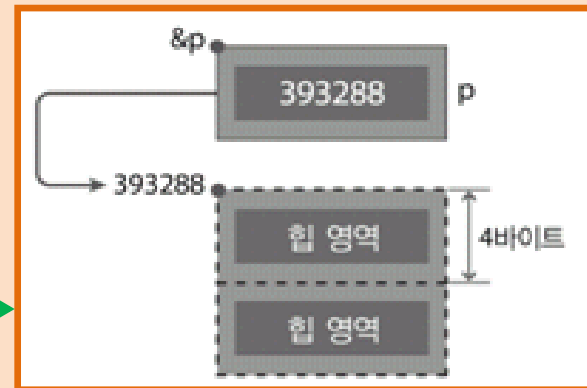
```
        printf("p[%d] : %d \n", i, p[i]);
```

```
    free(p);
```

```
    p=NULL;
```

```
    return 0;
```

```
}
```



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (13/22)

[4-7.c 실습]

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
```

```
{
    int i=0;
```

```
    int* p=(int*) malloc(sizeof(int)*2);
```

```
    p[0]=10;
```

```
    p[1]=20;
```

```
    p=(int*) realloc(p, sizeof(int)*1);
```

```
    p[0]=30;
```

```
    for(i=0; i<2; i++)
```

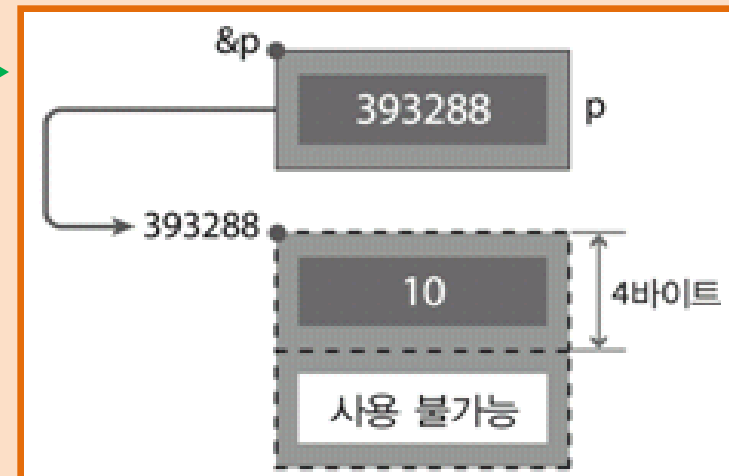
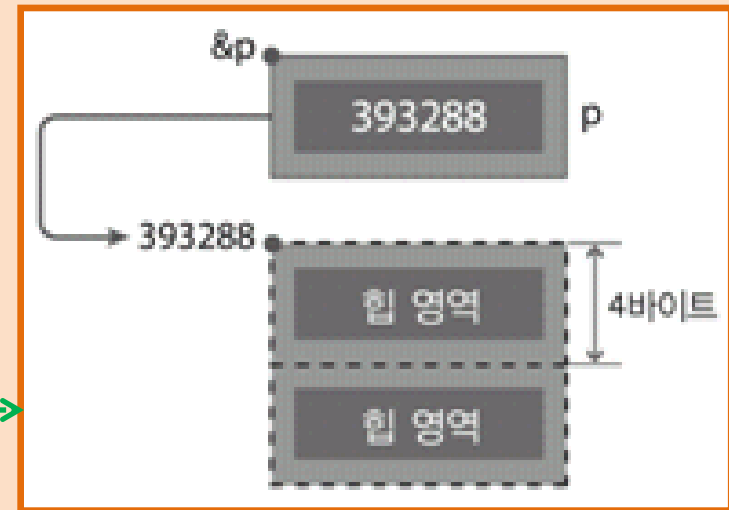
```
        printf("p[%d] : %d \n", i, p[i]);
```

```
    free(p);
```

```
    p=NULL;
```

```
    return 0;
```

```
}
```



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (14/22)

• 메모리 영역의 특징

특징	코드, 스택, 데이터 영역	힙영역
메모리 할당	컴파일 시간에 할당	런타임 시간(실행 시간)에 할당
메모리 해제	자동 해제	free() 함수로 해제
메모리 관리	컴파일러	프로그래머

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자

- 배울 내용

- ① malloc() 함수와 free() 함수

- ② calloc() 함수와 free() 함수

- ③ realloc() 함수와 free() 함수

- ④ 가변 인자

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (15/22)

- 가변 인자

- '함수의 인자 수를 고정하지 않는다.'

```
void add(int num, ... );
```

4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (16/22)

[4-8.c 실습]

```
#include <stdio.h>
void add (int num, ...);           // 가변 인자 함수 선언
int main(void)
{
    int a=10, b=20, c=30;

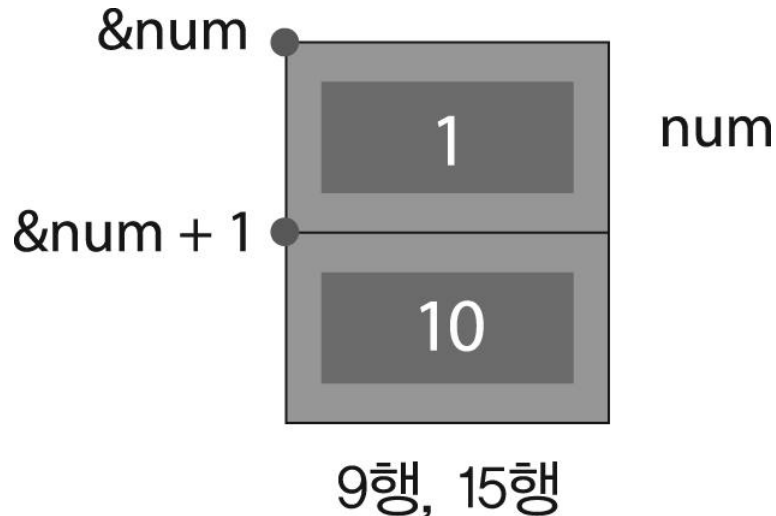
    9행 add(1, a);                  // 가변 인자 함수 호출 1
    10행 add(2, a, b);              // 가변 인자 함수 호출 2
    11행 add(3, a, b, c);           // 가변 인자 함수 호출 3
    return 0;
}

15행 void add(int num, ...)        // 가변 인자 함수 정의
{
    int* p=NULL;
    p=&num+1;

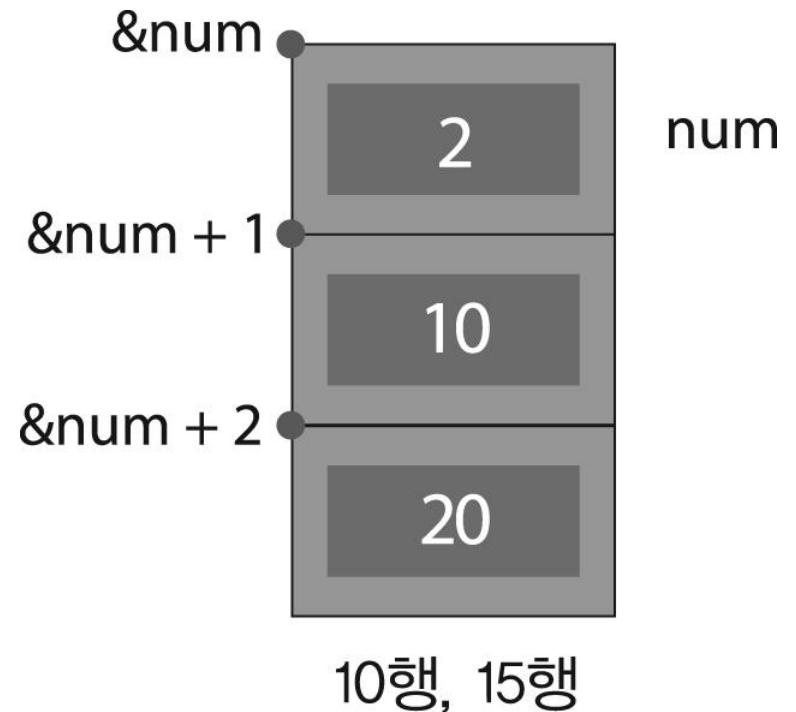
    if(num==1)
        printf("%d \n", p[0]);    // 10 출력
    else if (num==2)
        printf("%d \n", p[0]+p[1]); // 30 출력
    else
        printf("%d \n", p[0]+p[1]+p[2]); // 60 출력
}
```


4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (17/22)

- **add(1, a) 함수의 호출**

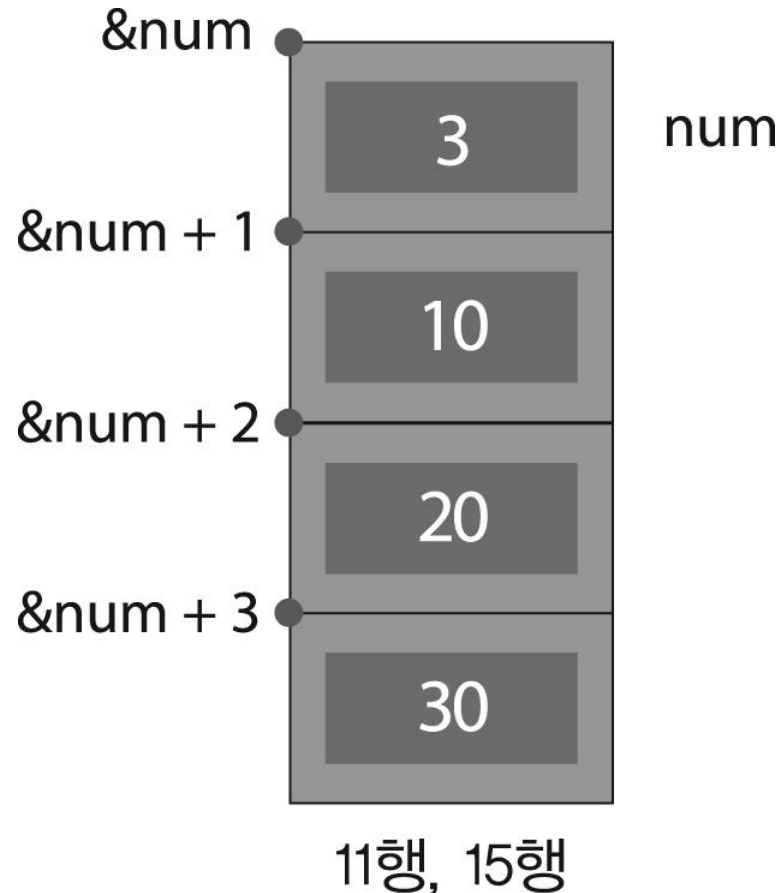


- ▶ **add(2, a, b) 함수의 호출**



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (18/22)

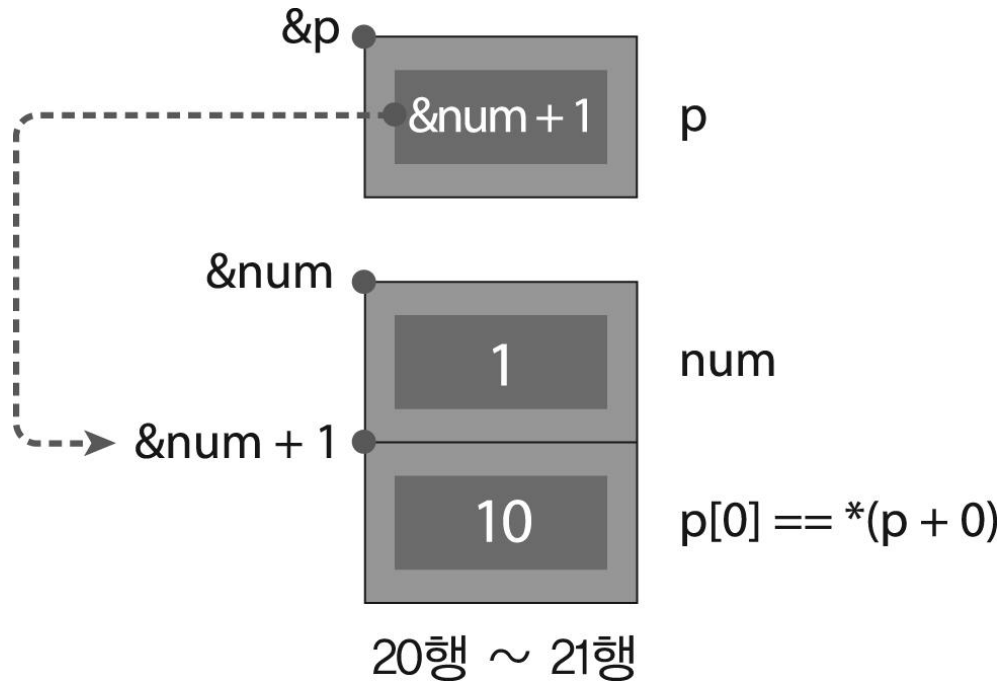
- **add(3, a, b, c) 함수의 호출**



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (19/22)

• $\text{num} == 1$ 인 경우

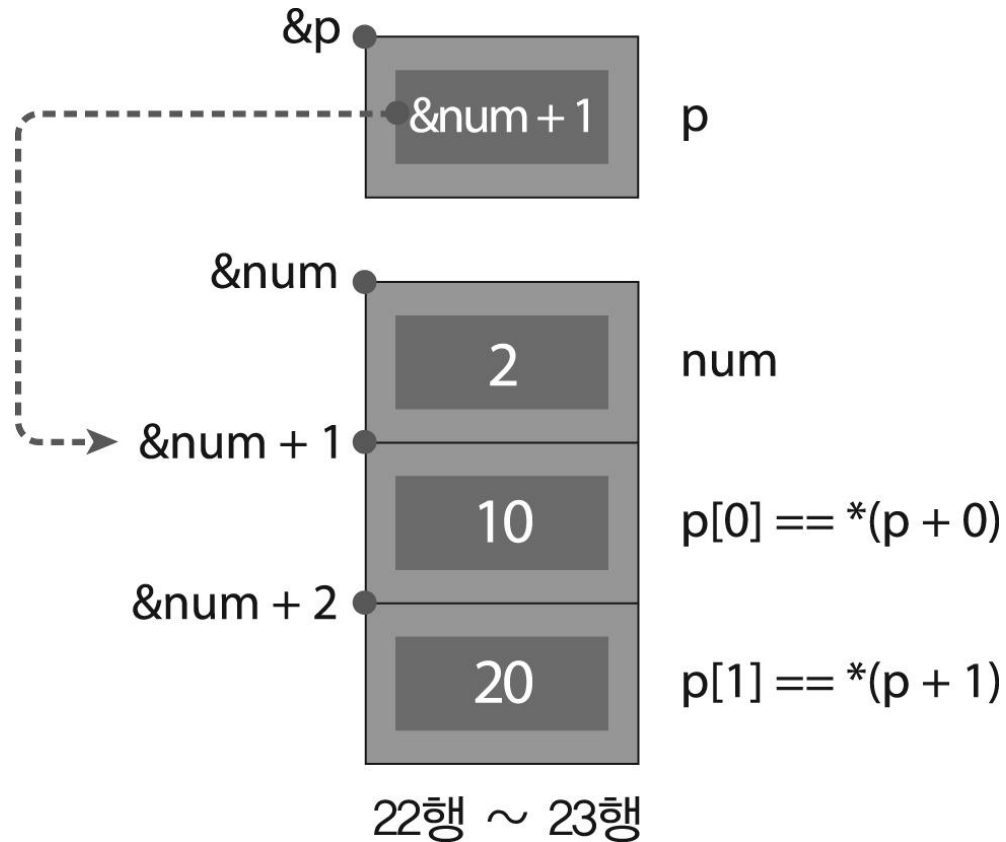
[4-8.c 분석]



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (20/22)

[4-8.c 분석]

• num==2인 경우



4.2 동적 메모리 할당 함수, 해제 함수 그리고 가변인자 (21/22)

[4-8.c 분석]

- **num==3인 경우**

