



C언어 (CSE2035)

(Chap9. Pointer Applications) (3-1)

Sungwon Jung, Ph.D.

Dept. of Computer Science and Engineering

Sogang University

Seoul, Korea

Tel: +82-2-705-8930

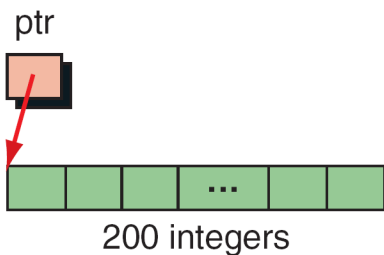
Email : jungsung@sogang.ac.kr

Contiguous Memory Allocation - calloc()

- 두 번째 메모리 관리 함수는 **calloc()** 이다.
- calloc() 함수의 프로토타입은 다음과 같다.

```
void *calloc (size_t element-count,
              size_t element_size);
```

- 할당 받은 메모리를 사용할 하나의 원소 크기와 전체 원소의 개수를 개별적인 인자로 받는다 .
- calloc() 함수는 메모리를 할당하면서 해당 메모리를 0으로 초기화 시킨다.



```
if (!(ptr = (int*)calloc (200, sizeof(int))
    // No memory available
    exit (100) ;

// Memory available
...
```

malloc()과 calloc()
의 차이점

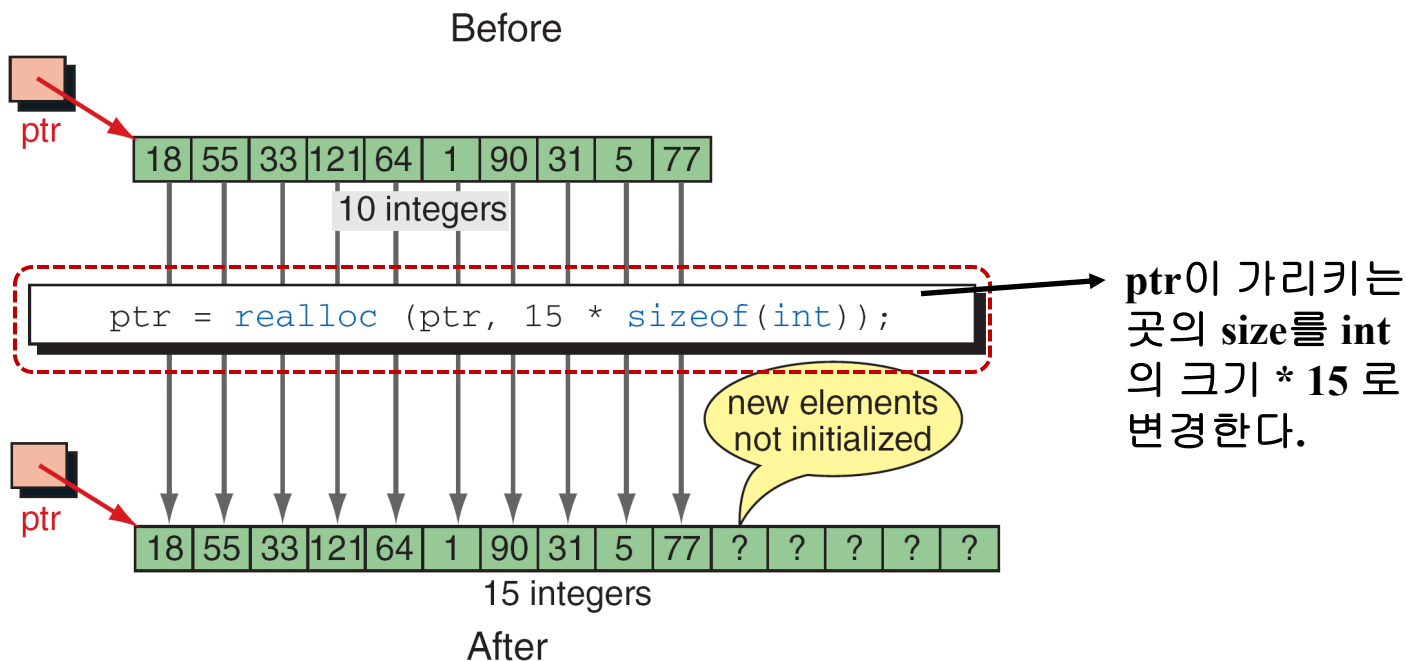
<calloc함수 호출 예>

Reallocation of Memory - realloc ()

- `realloc()` 함수는 변수의 메모리를 동적으로 변경하기 위하여 사용한다.
- 함수의 프로토타입은 다음과 같다.

```
void *realloc (void *ptr, size_t newSize);
```

- 즉, `ptr`이 현재 할당하고 있는 메모리의 크기를 `newSize`로 변경한다.



calloc() & realloc()

■ 예제 프로그램 - realloc 함수를 이용한 양수값 입력 프로그램

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int *ip;
7     int size = 5;
8     int cnt = 0;
9     int num;
10    int i;
11
12    ip=(int *)calloc(size, sizeof(int));
13    while(1){
14        printf("Input the positive integer : ");
15        scanf("%d", &num);
16        if(num<=0) break;
17        if(cnt<size) {
18            ip[cnt++]=num;
19        }
20        else {
21            size += 5;
22            ip=(int *)realloc(ip, size*sizeof(int));
23            ip[cnt++]=num;
24        }
25    }
26    for(i=0; i<cnt; i++) {
27        printf("%5d", ip[i]);
28    }
29    printf("\n");
30    free(ip);
31    return 0;
32 }

```

일정한 크기의 기억공간을 할당(calloc) 받고 나서, 입력되는 데이터가 양수일 경우에 기억공간을 재할당(realloc)한다.

할당 받은 기억공간이 남아있으면 데이터를 저장한다.

기억공간이 부족하면 크기를 늘려서 재할당 받는다.

```

[root@mclab chap10]# vi chap10-7.c
[root@mclab chap10]# gcc -o chap10-7 chap10-7.c
[root@mclab chap10]# ./chap10-7
Input the positive integer : 36
Input the positive integer : 28
Input the positive integer : 14
Input the positive integer : 0
      36   28   14
[root@mclab chap10]#

```



Releasing Memory - free ()

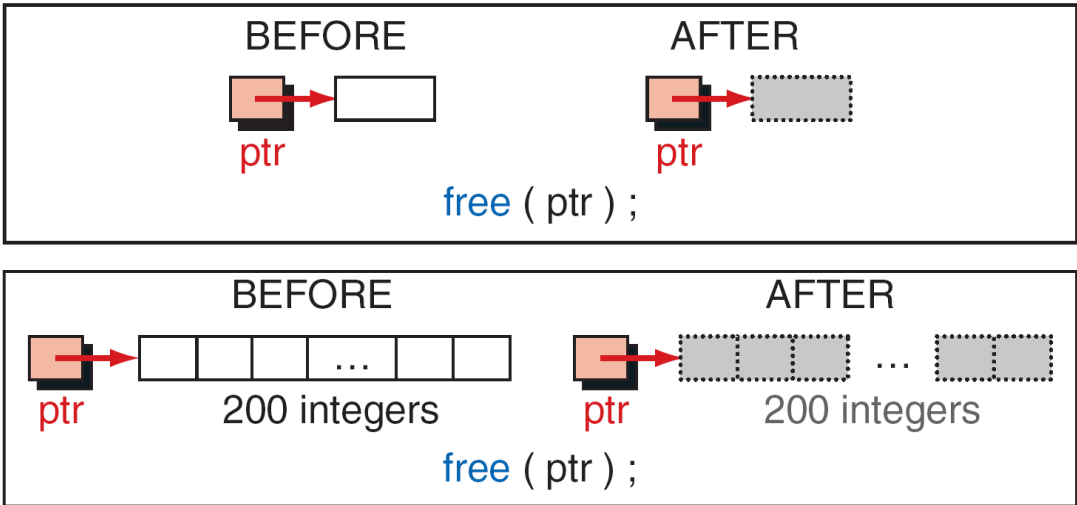
- 동적으로 할당 받은 메모리 영역에 대하여 반환하는 함수
 - free() 함수의 프로토타입은 다음과 같다.

```
void free (void *ptr);
```

- 사용하지 않는 메모리에 대한 계속적인 할당은 메모리 사용의 낭비의 단점이 있기 때문에 동적으로 할당 받은 필요 없는 메모리 영역에 대해서는 free()를 통해서 메모리를 반환하도록 한다.
- free() 는 ptr이 가리키는 기억 장소를 해제한다.
- 단, ptr이 NULL일 때는 아무 일도 하지 않는다.
- ptr은 malloc(), calloc(), realloc()에 의하여 이전에 할당되었던 기억 장소의 포인터이어야 한다.

Releasing Memory - free ()

- 다음은 free()의 동작을 보여주는 그림이다.



■ **Caution!**

동적으로 할당한 기억 장소 영역은 책임지고 해제하지 않으면 안 된다. 영역을 해제할 때까지는 반드시 포인터를 기억해 두어야 한다.

Releasing Memory - free ()

- 예제 프로그램 - 정수를 1개 할당, 해제하는 프로그램

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void main(void)
5 {
6     int *p;
7     p = (int *)malloc(1*sizeof(int));
8     if(p==NULL)
9         printf("Not allocated!\n");
10    else
11        free(p);
12 }
```

메모리 4byte를 할당
받음

제대로 할당되었는지
확인

메모리 4byte를 해제

Memory leak

■ 더 이상 사용되지 않는 메모리 공간의 반환

- 동적으로 할당 받은 메모리 공간은 더 이상 해당 공간을 참조하는 포인터가 없는 경우에도 heap 영역에 자리를 차지하고 있다.
- 동적 할당으로 생성된 메모리를 적절한 타이밍에 해제하지 않으면 해당 메모리는 생성된 만큼의 공간을 차지하면서도 정작 사용할 수 없는 상태가 된다

```
int *intArr;
```

```
intArr=(int *)malloc(sizeof(int)*5);
```

```
intArr=(int *)malloc(sizeof(int)*10);
```

```
free(intArr);
```

프로그램이 종료될 때 까지 남아있으나 사용할 수 없는 메모리

- 메모리 누수가 반복적으로 일어날 경우 시간이 지나면서 사용할 수 있는 메모리를 모두 사용하여 프로그램이 더 이상 올바르게 기능하지 못하게 된다.
- 이와 같은 문제를 메모리 누수(memory leak)라고 한다.

Dangling pointer

■ Dangling pointer

- 메모리가 삭제되거나 재 할당 될 때 유효하지 않은 메모리 공간을 참조하는 포인터.
- 해제된 공간을 참조하고 있거나 지역변수를 참조할 때 발생하기 쉽다.

```
int main(){
    int *intPointer;
    dangle(&intPointer);
    printf("%d", *intPointer);
    return 0;
}

void dangle(int **intPtr){
    int a=1;
    *intPtr=&a;
}
```

함수의 지역변수를 참조하고 있지만 함수 수행 종료 후 해당 변수는 **stack**에서 제거되기 때문에 **dangling pointer**가 된다

Dangling pointer

```
int main(){
    int *intPointer;
    intPointer=(int*)malloc(sizeof(int)*5);

    for(int i=0; i<5; i++)
        intPointer[i]=i;

    dangle(intPointer);

    printf("%d ", intPointer[1]);

    return 0;
}

void dangle(int *intPtr){
    free(intPtr);
}
```

이미 해제된 메모리 공간의 주소를 그대로 참조하고 사용하게 되면 프로그램이 오작동 하게 된다.

- Dangling pointer에 대한 접근은 프로그램의 오작동이나 종단을 유발하지만 컴파일 단계에서 검출되지 않기 때문에 주의가 필요하다.