



# **C Programming (CSE2035)** **(Chap9. Pointer Applications 2)**

---

**Sungwon Jung, Ph.D.**

**Bigdata Processing & DB LAB**  
**Dept. of Computer Science and Engineering**  
**Sogang University**  
**Seoul, Korea**  
**Tel: +82-2-705-8930**  
**Email : [jungsung@sogang.ac.kr](mailto:jungsung@sogang.ac.kr)**

## Passing an array to a function

- 배열을 인자로 하는 함수의 일반적인 호출 형태는 다음과 같다.
  - 배열의 시작 주소값을 인자로 전달한다.

```
doIt (aryName);
```

- 따라서 배열을 인자로 받는 함수의 헤더는 다음과 같을 것이다.

```
int doIt (int ary[])
```

- 또한 1차원 배열을 인자로 하는 경우 포인터를 이용한 경우도 가능하다.

```
int doIt (int *arySalary)
```

# Passing an array to a function

- 함수의 인자로 배열명을 전달하면, 배열의 시작 주소가 전달된다.

```
doIt (aryName);
```

배열명은 배열의 시작 주소값을 가지므로 주소가 전달된다.

- 호출되는 함수의 헤더에서는 형식 매개변수(formal parameter)로 포인터 변수를 사용하여야 한다.

```
int doIt (int *arySalary)
```

- 표기상의 편리성 때문에 포인터를 매개변수로 선언할 때 배열의 각괄호 표기법을 사용할 수 있다.
  - 이 때 배열로 선언된 형식 매개변수는 실질적으로는 포인터이다.
  - 배열 원소 자체는 복사되지 않는다.

```
int doIt (int ary[])
```

사실상 포인터!

# Passing an array to a function

- 예제 프로그램 – `a[]`와 `*a`가 같음을 보여주는 예제

```

1 #include <stdio.h>
2
3 double sum1(double a[], int n);
4 double sum2(double *a, int n);
5
6 int main(void)
7 {
8     double a[6]={1, 2, 3, 4, 5, 6};
9
10    printf("sum1 = %f\n", sum1(a,6));
11    printf("sum2 = %f\n", sum2(a,6));
12
13    return 0;
14 }
15
16 double sum1(double a[], int n)
17 {
18     int i;
19     double sum=0.0;
20     for(i=0; i<n; ++i)
21         sum+=a[i];
22     return sum;
23 }
24
25

```

배열의 주소값과 배열의  
길이를 인자로 전달

```

[root@mclab chap10]# ./chap10-4
sum1 = 21.000000
sum2 = 21.000000
[root@mclab chap10]#

```

두 함수의 수행 결과는 서로 같다.

형식 파라미터가 서로 다르다.

```

26 double sum2(double *a, int n)
27 {
28     int i;
29     double sum = 0.0;
30     for(i=0; i<n; i++)
31         sum+=a[i];
32     return sum;
33 }
34

```

## Passing an array to a function

- 배열을 인자로 하여 앞의 `sum()` 함수를 호출했을 때 계산되는 값은 다음 표와 같다. (`v`는 `main` 함수에서 선언된 배열이라 가정한다.)

호출	계산 및 리턴되는 값
<code>sum(v, 100)</code>	<code>v[0] + v[1] + ... + v[99]</code>
<code>sum(v, 88)</code>	<code>v[0] + v[1] + ... + v[87]</code>
<code>sum(&amp;v[7], k - 7)</code>	<code>v[7] + v[8] + ... + v[k - 1]</code>
<code>sum(v + 7, 2 * k)</code>	<code>v[7] + v[8] + ... + v[2 * k + 6]</code>

## Passing an array to a function

- 2차원 이상의 배열을 형식 매개변수로 갖는 함수는 (첫 번째를 제외한 나머지 차원의) 배열 크기를 알고 있어야 한다.

- 예) 다음은 3차원 배열을 형식 매개변수로 갖는 함수의 헤더이다.

```
float doIt (int bigAry[][12][5])
```

- 예) 다음은 4차원 배열을 형식 매개변수로 갖는 함수의 헤더이다.

- void func1(int a[][3][4][8]);

- 예) 다음의 함수 헤더는 컴파일 에러이다.

- int func2(int a[][]);

이 곳에 배열 크기를 써줘야 한다.

# Passing an array to a function

- 다음은 배열 원소의 값을 2배 하여 저장하는 `multiply()` 함수를 사용한 프로그램이다.

```
1  /* Read from keyboard & print integers multiplied by 2.
2      Written by:
3      Date:
4  */
5  #include <stdio.h>
6  #define SIZE  5
7
8  // Function Declarations
9  void multiply (int* pAry, int size);
10
11 int main (void)
12 {
13     // Local Declarations
14     int  ary [SIZE];
15     int* pLast;
16     int* pWalk;
17
18     // Statements
```

`multiply()` 함수의 선언부  
→ 배열을 인자로 받는다.

# Passing an array to a function

```

19  pLast = ary + SIZE - 1;
20  for (pWalk = ary; pWalk <= pLast; pWalk++)
21  {
22      printf("Please enter an integer: ");
23      scanf ("%d", pWalk);
24  } // for
25
26  multiply (ary, SIZE);
27
28  printf ("Doubled value is: \n");
29  for (pWalk = ary; pWalk <= pLast; pWalk++)
30      printf (" %3d", *pWalk);
31
32  return 0;
33  } // main
34

```

pLast → 배열의 마지막 요소를 참조한다.

배열 초기화 → scanf()를 통해 입력 받는다.

multiply() 호출

출력



# Passing an array to a function

```

35  /* ===== multiply =====
36      Multiply elements in an array by 2
37      Pre   array has been filled
38      size indicates number of elements in array
39      Post  values in array doubled
40  */
41  void multiply (int* pAry, int size)
42  {
43      // Local Declarations
44      int* pWalk;
45      int* pLast;
46
47      // Statements
48      pLast = pAry + size - 1;
49      for (pWalk = pAry; pWalk <= pLast; pWalk++)
50          *pWalk = *pWalk * 2;
51      return;
52  } // multiply
53  // ===== End of Program =====

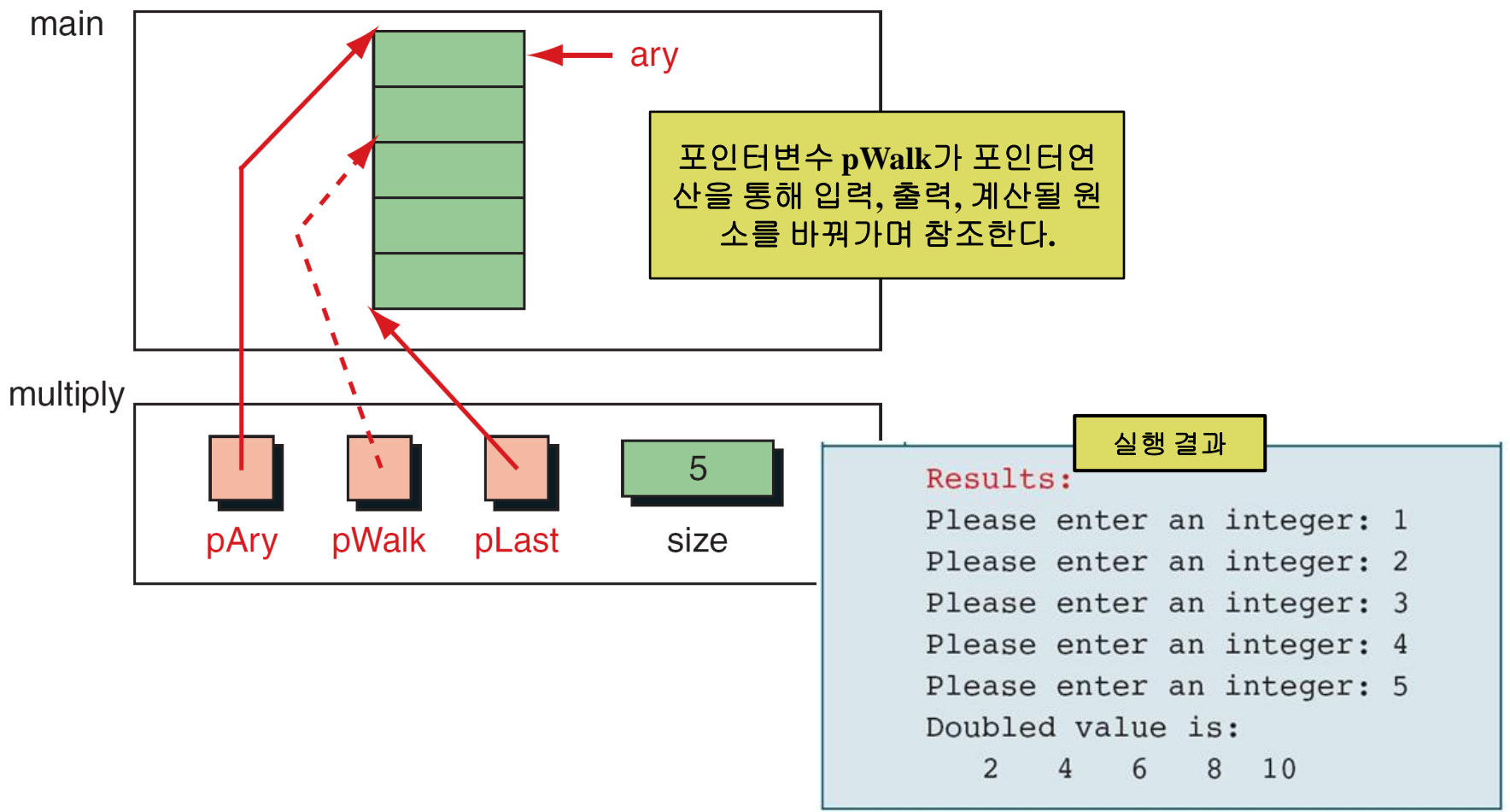
```

배열과 배열크기를 인수로 받는다.

배열 원소 값 \* 2

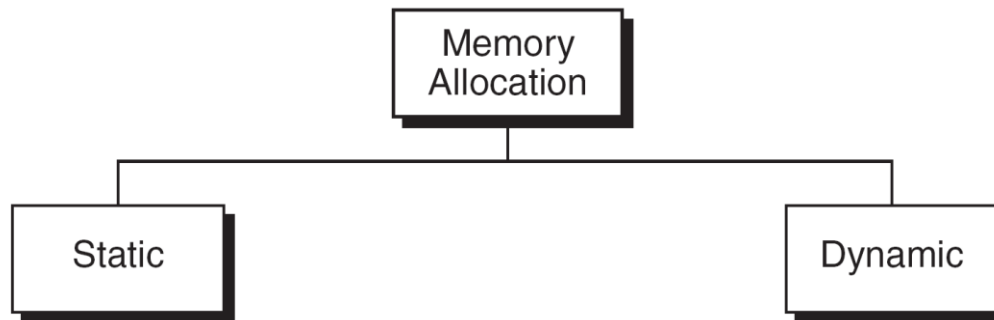
# Passing an array to a function

- 다음은 `multiply()` 함수를 사용한 프로그램의 구조이다.



# Memory allocation functions

- C언어를 사용하면서 변수 영역의 메모리를 할당 받는 것으로 다음의 두 가지 방법이 가능하다.



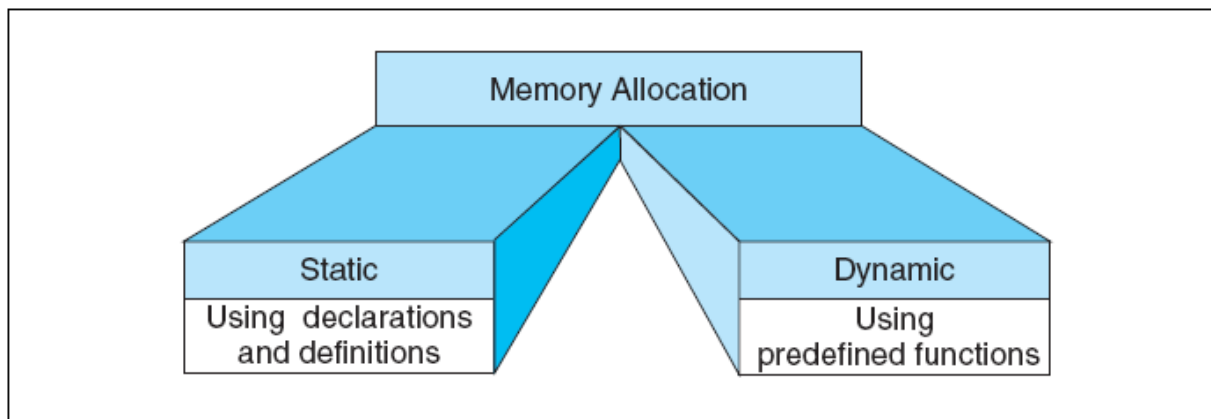
- **Static memory allocation**

- 정적 메모리 할당은 소스프로그램 안에서 메모리의 선언과 정의가 완벽하게 명시되어 있어야 한다.
- 실행 시간 동안에 할당된 byte 수 등이 변할 수 없다.
  - 입력되는 데이터 길이에 대해 정확하게 알고 있어야만 사용 가능하다.

# Memory allocation functions

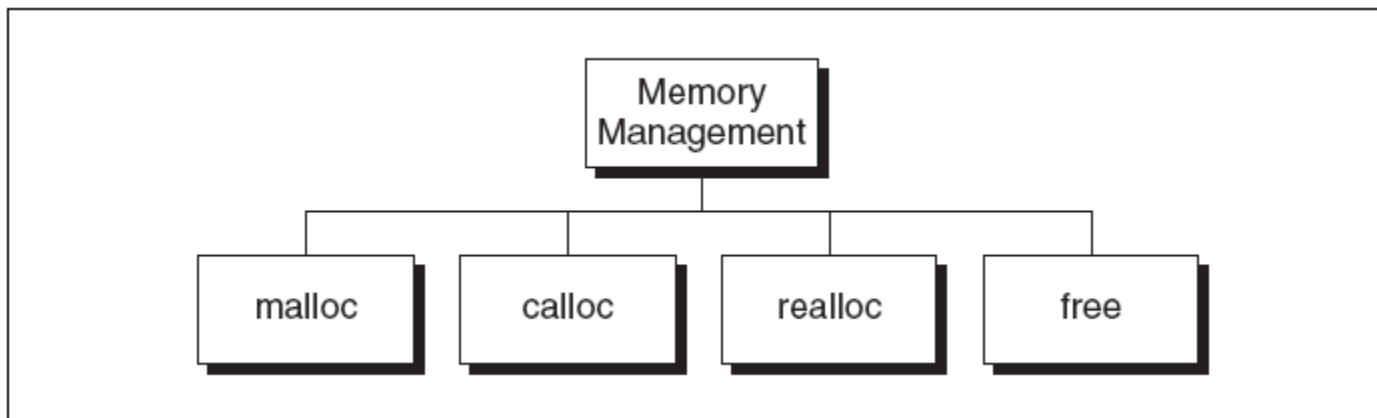
## ■ 동적 메모리 할당

- 메모리의 낭비를 최소화하기 위해 프로그램의 실행 중에 입력되는 데이터에 맞게 기억공간이 할당되어야 한다. ➔ 메모리의 동적 할당
- 메모리의 동적 할당과 해제 기능은 라이브러리로 구현되어 있고, 이를 이용하기 위해서는 반드시 포인터를 이용한다.
  - 동적 할당이란 프로그램 실행 중에 ,즉 `run time`때 메모리 영역을 할당한다는 뜻이다.



# Memory allocation functions

- Memory management functions



- 메모리의 동적 할당에 사용되는 메모리 할당 함수들
  - malloc(), calloc(), realloc()은 메모리를 할당하기 위한 함수
  - free()는 더 이상 사용하지 않는 메모리를 리턴하는 함수
- 함수를 사용하기 위해서는 “stdlib.h” 헤더파일을 포함해야 한다.
  - #include<stdlib.h>



# malloc()

- **malloc()** 함수는 메모리를 할당하기 위한 함수이다.
- 함수의 프로토타입은 다음과 같다.

```
void *malloc (size_t size);
```

- malloc() 함수는 할당한 메모리의 시작주소를 void 포인터로써 리턴한다. 할당에 실패한 경우 NULL을 되돌려 준다.
- malloc() 은 크기가 size인 객체(기억 장소)를 할당한다.
- size\_t 타입은 stdio.h에서 정의된 타입으로 unsigned integer의 값을 저장한다.

## malloc()

- 다음은 malloc() 함수의 일반적인 사용 예이다.

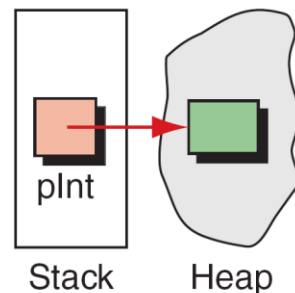
```
pInt = (int *)malloc (sizeof (int));
```

int 타입 포인터

- pInt에는 하나의 int형 데이터를 저장할 수 있는 크기의 메모리가 할당
- n개의 int 형 원소를 저장할 수 있는 메모리를 할당 받기 위해서는
  - scanf(“%d”, &n);
  - malloc(sizeof(int) \* n)
- 만일 malloc() 함수가 메모리 영역을 할당하는데 실패한다면 NULL 포인터를 리턴할 것이다.

```
if (!(pInt = malloc(sizeof(int))))
    // No memory available
    exit (100) ;
// Memory available
...
```

<malloc함수 호출 예>



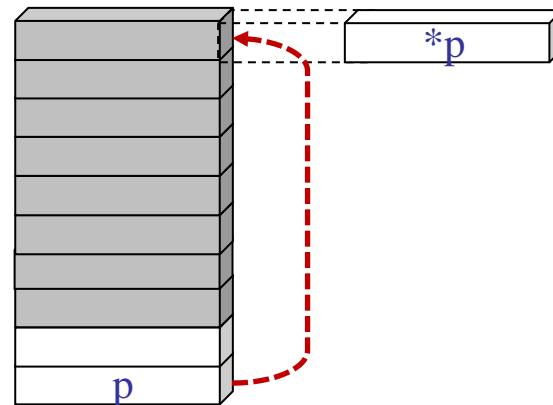
# Dynamic allocation of array

## ■ Dynamic allocation of array

```
int *p;  
p = (int *)malloc(8 * sizeof(int));
```

- malloc은 8개의 int를 위한 기억 장소를 할당해 그 시작 주소값을 되돌려 준다.

malloc 함수로  
할당된 영역



- 포인터 변수가 있으면 포인터가 가리키는 영역을 배열처럼 사용가능
- 포인터 변수 p가 가리키는 원소의 i번째 원소를 나타내는 표현식
  - $*(p+i)$ ,  $p[i]$



# malloc()

## ■ 예제프로그램 - malloc을 이용한 변수 할당

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

```
int main(void)
{
```

```
    int *p;
```

```
    p=(int*)malloc(1*sizeof(int));
```

```
    if(p==NULL) {
```

```
        printf("not allocated\n");
        return 1;
```

```
    }
```

```
    *p=1;
```

```
    printf("%d\n", *p);
    return 0;
```

```
}
```

```
[root@mclab chap10]# ./chap10-5
```

```
1
```

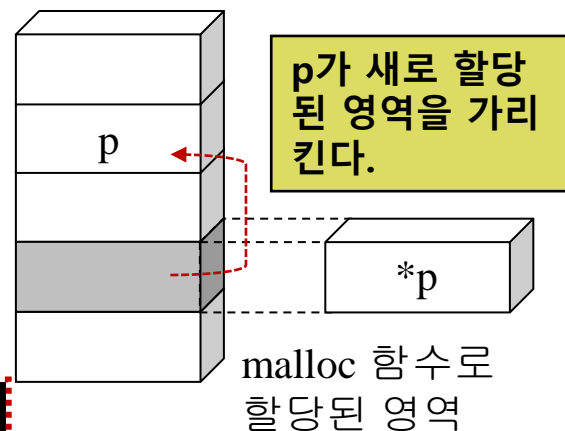
```
[root@mclab chap10]#
```

int형 1개 크기의 메모리 할당하고 그 주소값을 p에 넘겨준다.

새로 할당된 메모리영역에 접근

```
int *p;
int a;
p = &a;
*p = 3;
printf("%d", *p);
```

```
int *p;
p = (int *)malloc(sizeof(int))
*p = 3;
printf("%d", *p);
```



# malloc()

- 예제프로그램 - 메모리를 동적으로 할당 받아 배열로 사용하는 프로그램

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int *ip;
7     int i, sum = 0;
8
9     ip = (int *)malloc(5*sizeof(int));
10    if(ip==0) {
11        printf("short memory\n");
12        return 1;
13    }
14    printf("Input the age of 5 people : ");
15    for(i=0; i<5; i++) {
16        scanf("%d", ip+i);
17        sum+=ip[i];
18    }
19    printf("Average : %.11f\n", sum/5.0);
20    free(ip);
21    return 0;
22 }
```

20바이트의  
기억공간  
할당

```
[root@mclab chap10]# vi chap10-6.c
[root@mclab chap10]# gcc -o chap10-6 chap10-6.c
[root@mclab chap10]# ./chap10-6
Input the age of 5 people : 21 27 24 22 35
Average : 25.8
[root@mclab chap10]#
```

할당 받은 메모리를  
배열처럼 사용하여  
값을 입력 받는 부  
분