

제 16 장 동적 메모리

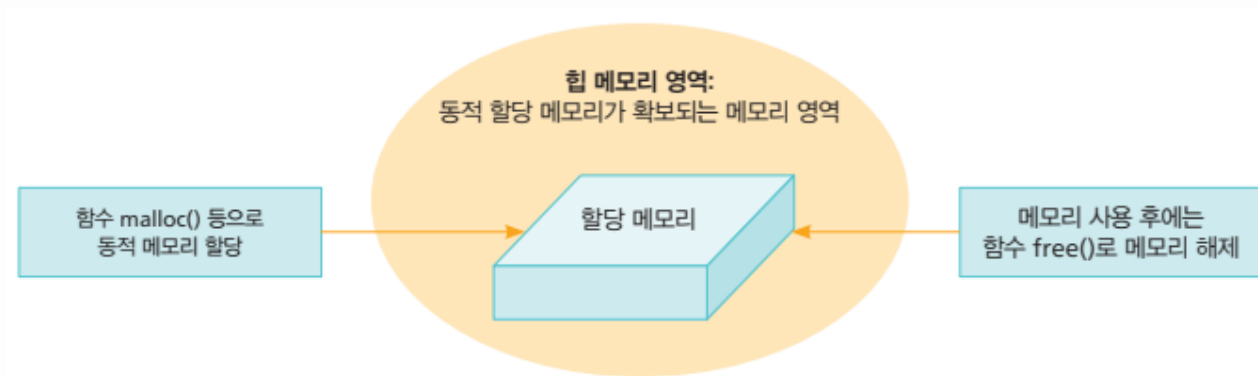
동적 메모리 할당

동적 메모리 할당 방식

```
int* pi = NULL;  
  
pi = (int*) malloc( sizeof(int) ); //동적 메모리 할당  
*pi = 7; //동적 메모리에 내용 값 7 저장
```

동적 메모리 관련 함수

메모리 연산	기본값	함수 원형	기능
메모리 할당	없음	<code>void * malloc(size_t)</code>	인자만큼의 메모리 할당 후 기본 주소 반환
	0	<code>void * calloc(size_t , size_t)</code>	뒤 인자 만큼의 메모리 크기로 앞 인자 수 만큼 할당 후 기본 주소 반환
기존 메모리 변경	이전 값	<code>void * realloc(void *, size_t)</code>	앞 인자의 메모리를 뒤 인자 크기로 변경 후, 기본 주소 반환
메모리 해제	해당 없음	<code>void free(void *)</code>	인자를 기본 주소로 갖는 메모리 해제



함수 malloc()

함수 malloc() 함수원형

자료형 size_t는 자료형의 크기를 의미한다.

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

```
int *pi = (int *) malloc( sizeof(int) );  
*pi = 3;
```

반환값은 이 값을 받는 자료유형의 포인터로
변환하여 포인터 변수에 저장된다.

함수 malloc()의 인자는 할당할 변수의 크기를
sizeof 연산자를 이용하여 지정한다.

함수 free()

함수 free() 함수원형

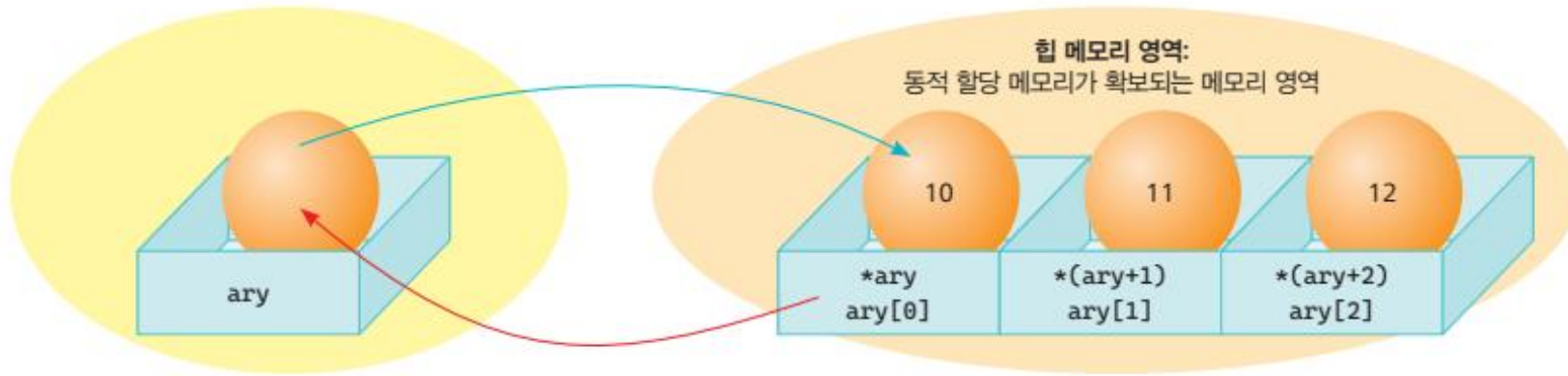
```
void free(void *);
```

```
free(pi);
```

함수 malloc()에 의한 배열 공간 할당

```
int *ary;  
ary = (int *) malloc( sizeof(int)*3 );  
ary[0] = 10; ary[1] = 11; ary[2] = 12;
```

함수 malloc()에 의해 할당된 저장공간은 int형 3개이며 주소 값을 ary에 저장하고 있다. ary[i]로 각 원소를 참조할 수 있다.



```

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

int main(void) {
    int* p = (int*)malloc(sizeof(int) * 5);

    if (p == NULL) {
        printf("메모리 할당 실패\n");
        return 1;
    }

    for (int i = 0; i < 5; i++) {
        p[i] = (i + 1) * 10;
    }

    for (int i = 0; i < 5; i++) {
        printf("p[%d] = %d\n", i, p[i]);
    }

    free(p);
    return 0;
}

```

Microsoft Visual Studio 디버거

```

p[0] = 10
p[1] = 20
p[2] = 30
p[3] = 40
p[4] = 50

```

C:\Users\user\Desktop
이 창을 닫으려면 아무

예제

입력할 점수의 개수를 입력 >> 5

5개의 점수 입력 >> 89 70 67 92 99

입력 점수: 89 70 67 92 99

합: 417 평균: 83.4


```

int n = 0;
printf("입력할 점수의 개수를 입력 >> ");
scanf("%d", &n);

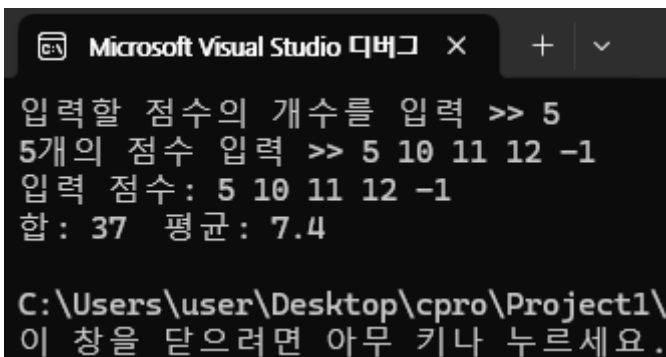
int* ary = NULL;
if ((ary = (int*)malloc(sizeof(int) * n)) == NULL) {
    printf("메모리 할당에 문제가 있습니다.");
    exit(1);
}

printf("%d개의 점수 입력 >> ", n);
int sum = 0;
for (int i = 0; i < n; i++) {
    scanf("%d", (ary + i));
    sum += *(ary + i);
}
printf("입력 점수: ");
for (int i = 0; i < n; i++)
    printf("%d ", *(ary + i));
printf("\n");

printf("합: %d  평균: %.11f\n", sum, (double)sum / n);

free(ary);
return 0;

```



Microsoft Visual Studio 디버그 × + ▾

```

입력할 점수의 개수를 입력 >> 5
5개의 점수 입력 >> 5 10 11 12 -1
입력 점수: 5 10 11 12 -1
합: 37  평균: 7.4

C:\Users\user\Desktop\cpro\Project1\
이 창을 닫으려면 아무 키나 누르세요.

```

함수 calloc()

함수 calloc() 함수원형

```
void * calloc(size_t num, size_t size);
```

할당될 메모리 항목의 개수이다.

할당될 메모리 한 원소의 크기이다.

- 함수 calloc()은 원소 크기가 size인 배열크기 num개의 공간을 할당하여 포인터를 반환하고 원소값은 모두 0으로 저장

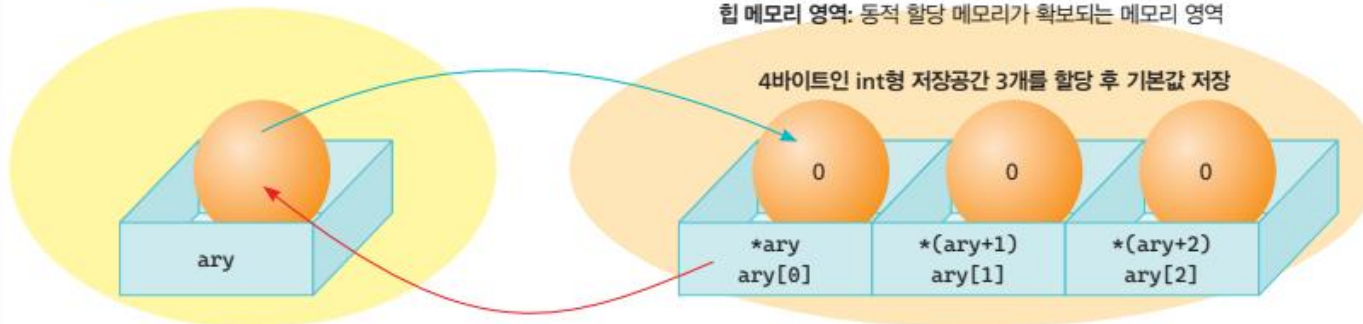
```
int *ary = NULL;
```

```
ary = (int *) calloc( 3, sizeof(int) );
```

반환값은 이 값을 받는 자유유형의 포인터로
변환하여 포인터 변수에 저장된다.

힙 메모리 영역: 동적 할당 메모리가 확보되는 메모리 영역

4바이트인 int형 저장공간 3개를 할당 후 기본값 저장



함수 calloc()에 의해 할당된 저장공간은 int형 3개이며 주소값을
ary에 저장하고 있다. ary[i]로 각 원소를 참조할 수 있다.

```
int *arr1 = (int *)malloc(5 * sizeof(int));
int *arr2 = (int *)calloc(5, sizeof(int));

for (int i = 0; i < 5; i++) {
    printf("%d ", arr1[i]);
}
printf("\n");

for (int i = 0; i < 5; i++) {
    printf("%d ", arr2[i]);
}
printf("\n");

free(arr1);
free(arr2);
```

함수 realloc()

함수 realloc() 함수원형

```
void * realloc(void *p, size_t size);
```

할당되는 총 메모리 크기이다.

- 함수 realloc()은 이미 확보한 메모리 p를 다시 지정한 크기 size로 변경하는 함수이며, 이미 확보한 p가 NULL이면 malloc()과 같은 기능을 수행

사용자가 입력하는 정수들을 동적으로 저장하는 프로그램을 작성하시오. 입력된 정수는 -1이 입력될 때까지 계속 받아들인다. 정수가 저장될 배열은 `malloc()` 과 `realloc()` 을 이용하여 필요 시 자동으로 크기를 2 배씩 확장한다. 마지막에 입력된 모든 정수를 출력한다.

```
C:\Users\Wuser\source\Wrepo. x + v
Enter numbers (exit with -1):
Input: 1
Input: 2
Input: 3
Input: 4
Input: 9
Input:
```

```

int* arr = NULL;
int size = 0, capacity = 2;

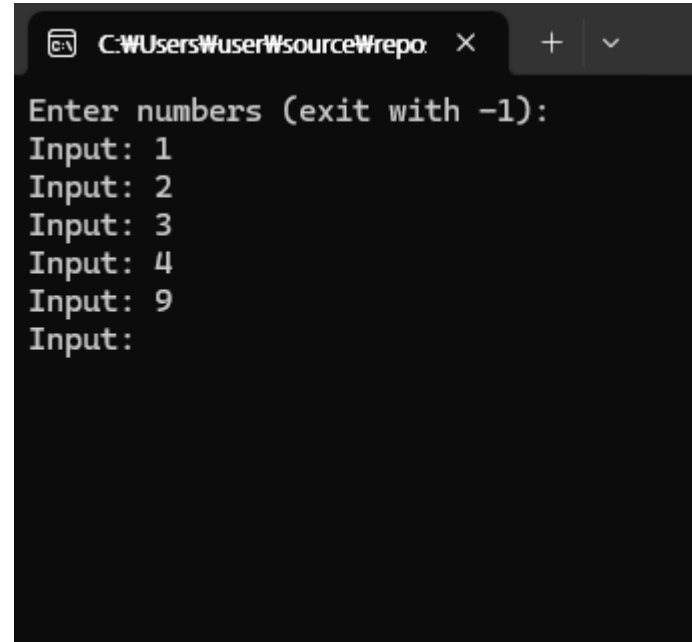
arr = (int*)malloc(capacity * sizeof(int));
if (arr == NULL) return 1;

printf("Enter numbers (exit with -1):\n");
while (1) {
    int num;
    printf("Input: ");
    scanf("%d", &num);

    if (num == -1) break;
    if (size == capacity) {
        capacity *= 2;
        int* temp = realloc(arr, capacity * sizeof(int));
        if (temp == NULL) {
            free(arr);
            return 1;
        }
        arr = temp;
    }
    arr[size++] = num;
}

printf("Entered numbers: ");
for (int i = 0; i < size; i++) {
    printf("%d ", arr[i]);
}
printf("\n");
free(arr);
return 0;

```



```

C:\Users\Wuser\source\Wrepo x + v
Enter numbers (exit with -1):
Input: 1
Input: 2
Input: 3
Input: 4
Input: 9
Input:

```

2차원배열 동적 할당

→ 포인터 배열 사용

1. 각 행을 가리키는 포인터 배열 할당
2. 각 행에 대해 메모리 할당

```

1  ✓ #include <stdio.h>
2    #include <stdlib.h>
3
4  ✓ int main() {
5      int rows = 3;
6      int cols = 4;
7      int i, j;
8
9      int **array = (int **)malloc(rows * sizeof(int *));
10  ✓ if (array == NULL) {
11      |     fprintf(stderr, "Memory allocation failed for rows\n");
12      |     return 1;
13      | }
14
15  ✓ for (i = 0; i < rows; i++) {
16      |     array[i] = (int *)malloc(cols * sizeof(int));
17      | }
18
19  ✓ for (i = 0; i < rows; i++) {
20  ✓     for (j = 0; j < cols; j++) {
21         |         array[i][j] = i * cols + j;
22         |     }
23     }

```

```

25     printf("2D Array Elements:\n");
26     for (i = 0; i < rows; i++) {
27         |     for (j = 0; j < cols; j++) {
28             |         printf("%d ", array[i][j]);
29             |     }
30         |     printf("\n");
31     }
32
33     // 메모리 해제
34     for (i = 0; i < rows; i++) {
35         |     free(array[i]); // 각 행 해제
36     }
37     free(array); // 포인터 배열 해제
38
39     return 0;
40 }

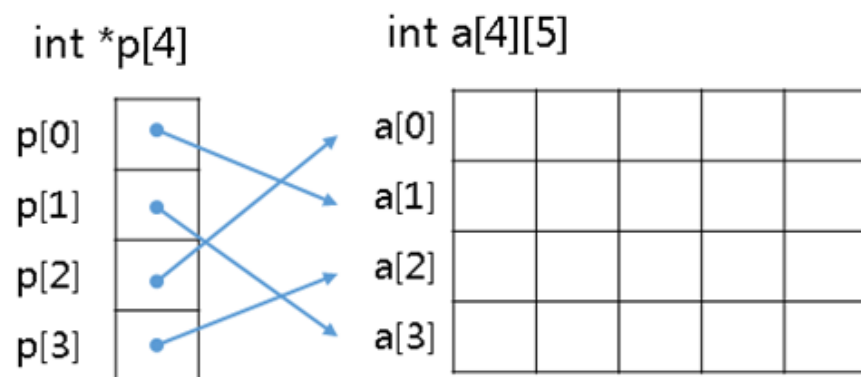
```

```

Microsoft Visual Studio 디버그
2D Array Elements:
0 1 2 3
4 5 6 7
8 9 10 11
C:\Users\user\Desktop\cpro\Project

```


4. 파일에서 데이터를 읽어 2차원 배열에 저장하고 p 배열을 사용해서 배열 a의 원소를 출력하는 코드를 작성하시오.



- 각 1차원 포인터 배열 `int *p[]`를 선언하고, 각 포인터가 a의 행을 가르키도록 설정한다.
(주어진 순서대로 `p[0]~p[3]`에 `a[2]`, `a[3]`, `a[0]`, `a[1]`을 저장)
- 출력은 함수 `print1DArray`를 사용하여 `p[0]`부터 `p[3]`까지 각 배열을 순서대로 출력한다.

`void print1DArray(int *arr, int n);`

입력(f4.txt)	출력
10 30 5 1 7	20 16 9 2 2
20 16 9 2 2	4 5 7 8 5
8 19 22 3 3	10 30 5 1 7
4 5 7 8 5	8 19 22 3 3