



5주차. #include <C\_language.h>

---

# Whois 신입회원 교육

Whois 박상현





## 목 차

---

### 1. 배열의 확장

다차원 배열  
배열 이용하기

### 2. 구조체

포인터 사용하기  
배열과 포인터의 관계  
Call by ~~~

### 3. 동적할당

포인터 사용하기  
배열과 포인터의 관계  
Call by ~~~

### 4. 그 외 +a

포인터 사용하기  
배열과 포인터의 관계  
Call by ~~~



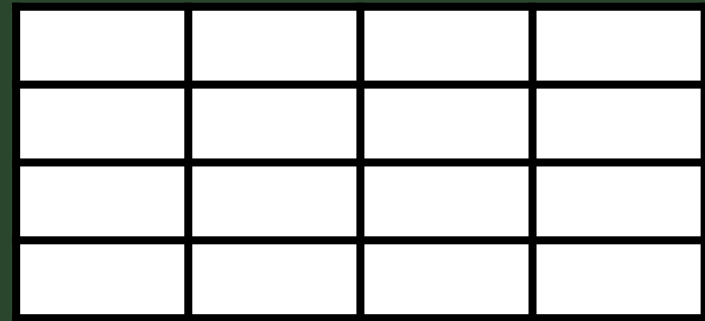
# 1. 배열과 포인터

# 1. 다차원 배열

---



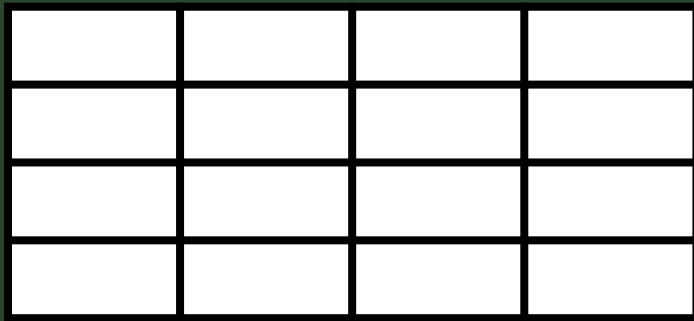
1차원 배열



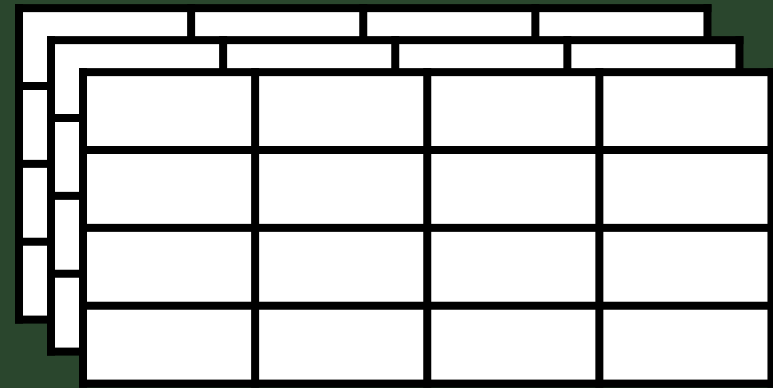
2차원 배열

# 1. 다차원 배열

---




2차원 배열




N차원 배열

## 2. 2차원 배열 사용하기

---

```
arr[N-1][M-1] = 20; // N행(세로), M열(가로)의 위치에 정수 20을 저장  
Printf("%d", arr[N-1][M-1]); // N행, M열의 위치에 저장된 값 출력
```

## 2. 2차원 배열 사용하기

---

```
arr[3][3] = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9}  
};
```

```
arr[3][3] = {  
    {1},  
    {4, 5},  
    {7, 8, 9}  
};
```

```
arr[3][3] = {1, 2, ,3 ,4, 5, 6, 7};
```

```
arr[][4] = {1, 2, ,3 ,4, 5, 6, 7, 8};
```

## 2. 2차원 배열 사용하기

---

`arr[0][1] = 2;`

0	1	0	0
0	0	0	0
0	0	0	0
0	0	0	0



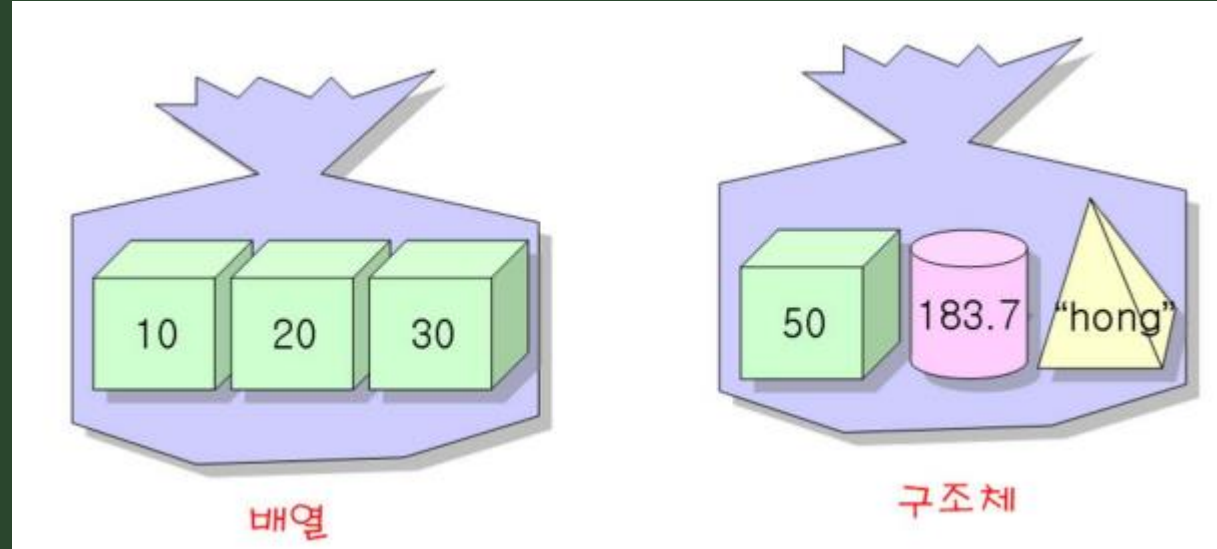
## 2. 2차원 배열 사용하기

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  int main() {
5      int villa[4][2];
6      int popu, i, j;
7
8      for (i = 0; i < 4; i++) {
9          for (j = 0; j < 2; j++) {
10             printf("%d층 %d호 인구수: ", i + 1, j + 1);
11             scanf("%d", &villa[i][j]);
12         }
13     }
14
15     for (i = 0; i < 4; i++) {
16         popu = 0;
17         popu += villa[i][0];
18         popu += villa[i][1];
19         printf("%d층 인구수: %d\n", i + 1, popu);
20     }
21     return 0;
22 }
```

1층	1호	인구수:	2
1층	2호	인구수:	6
2층	1호	인구수:	4
2층	2호	인구수:	3
3층	1호	인구수:	2
3층	2호	인구수:	4
4층	1호	인구수:	3
4층	2호	인구수:	5
1층	인구수:	8	
2층	인구수:	7	
3층	인구수:	6	
4층	인구수:	8	

## 2. 구조체

# 1. 구조체



# 1. 구조체

---

```
struct person // person이라는 이름의 구조체 정의
{
    char name[20]; // 이름 저장을 위한 멤버
    char phoneNum[20]; // 전화번호 저장을 위한 멤버
    int age; // 나이 저장을 위한 멤버
};
```

# 1. 구조체

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5  struct person {
6      char name[20];
7      char phoneNum[20];
8      int age;
9  };
10
11 int main() {
12     struct person man1, man2;
13
14     strcpy(man1.name, "박상현");
15     strcpy(man1.phoneNum, "010-5882-7880");
16     man1.age = 21;
17
18     printf("이름 입력: "); scanf("%s", man2.name);
19     printf("번호 입력: "); scanf("%s", man2.phoneNum);
20     printf("나이 입력: "); scanf("%d", &(man2.age));
21
22     printf("이름: %s\n", man1.name);
23     printf("번호: %s\n", man1.phoneNum);
24     printf("나이: %d\n", man1.age);
25
26     printf("이름: %s\n", man2.name);
27     printf("번호: %s\n", man2.phoneNum);
28     printf("나이: %d\n", man2.age);
29
30     return 0;
31 }
```

```
이름 입력: 손흥민
번호 입력: 010-7777-7777
나이 입력: 31
이름: 박상현
번호: 010-5882-7880
나이: 21
이름: 손흥민
번호: 010-7777-7777
나이: 31
```

# 1. 구조체

```
1  #include <stdio.h>
2
3  struct point {
4      int xpos;
5      int ypos;
6  };
7
8  typedef struct point Point;
9
10 typedef struct person {
11     char name[20];
12     char phoneNum[20];
13     int age;
14 } Person;
15
16 int main() {
17     Point pos = { 10, 20 };
18     Person man = { "이승기", "010-1212-0001", 21 };
19     printf("%d %d\n", pos.xpos, pos.ypos);
20     printf("%s %s %d\n", man.name, man.phoneNum, man.age);
21
22     return 0;
23 }
```

```
10 20
이승기 010-1212-0001 21
```

## 2. 구조체 배열

---

stru [0][0]	stru [0][1]	stru [0][2]	stru [0][3]
stru [1][0]	stru [1][1]	stru [1][2]	stru [1][3]
stru [2][0]	stru [2][1]	stru [2][2]	stru [2][3]
stru [3][0]	stru [3][1]	stru [3][2]	stru [3][3]

## 2. 구조체 배열

---

```
struct person arr[3] = {  
    {"이승기", "010-1212-0001", 21}, // 첫 번째 요소의 초기화  
    {"정지영", "010-1313-0002", 22}, // 두 번째 요소의 초기화  
    {"한지수", "010-1717-0003", 19}, // 세 번째 요소의 초기화  
};
```



## 2. 구조체 배열

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  struct point {
5      int xpos;
6      int ypos;
7  };
8
9
10 int main() {
11     struct point arr[3];
12
13     for (int i = 0; i < 3; i++) {
14         printf("점의 좌표를 입력: ");
15         scanf("%d %d", &arr[i].xpos, &arr[i].ypos);
16     }
17
18     for (int i = 0; i < 3; i++) {
19         printf("[%d, %d] ", arr[i].xpos, arr[i].ypos);
20     }
21
22     return 0;
23 }
```

점의 좌표	입력	:	2	4
점의 좌표	입력	:	3	8
점의 좌표	입력	:	8	9
[2, 4] [3, 8] [8, 9]				

### 3. 구조체 변수와 포인터

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  struct point {
5      int xpos;
6      int ypos;
7  };
8
9
10 int main() {
11     struct point pos1 = { 1, 2 };
12     struct point pos2 = { 100, 200 };
13     struct point* pptr = &pos1;
14
15     (*pptr).xpos += 4;
16     (*pptr).ypos += 5;
17     printf("[%d, %d]\n", pptr->xpos, pptr->ypos);
18
19     pptr = &pos2;
20     pptr->xpos += 1;
21     pptr->ypos += 2;
22     printf("[%d, %d]\n", (*pptr).xpos, (*pptr).ypos);
23
24     return 0;
25 }
```

[5, 7]  
[101, 202]

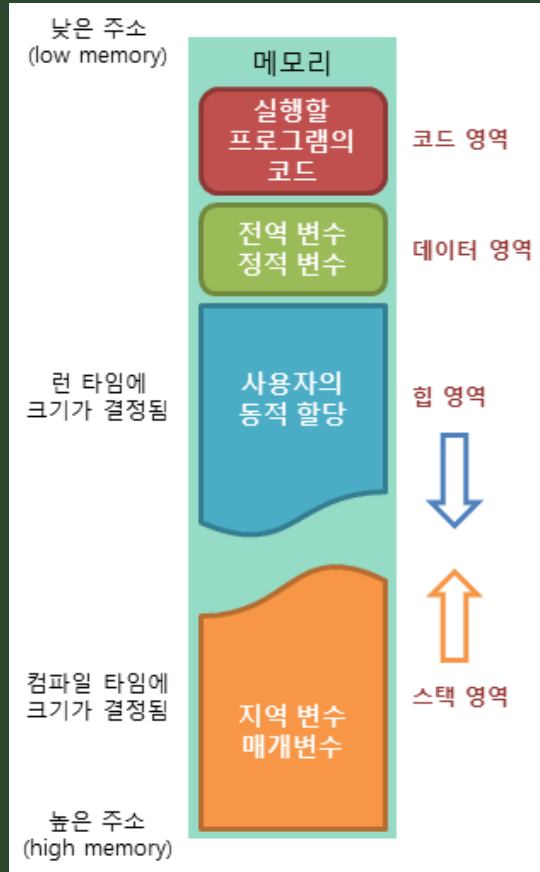
### 3. 동적할당

# 1. 동적할당

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  int main() {
5      int num;
6
7      scanf("%d", &num);
8
9      int arr[num];
10
11     return 0;
12 }
```

왼쪽에 코드를 사용할 경우, 컴파일 오류  
(gcc에서는 사용가능)

# 1. 동적할당



- 코드 영역(Code Area)  
: 실행할 프로그램의 코드가 저장되는 메모리 공간
- 데이터 영역(Data Area)  
: 전역 변수, static 변수 할당, 프로그램 종료 시까지 존재
- 스택 영역(Stack Area)  
: 지역 변수 및 매개 변수, 수시로 생성/소멸
- 힙 영역(Heap Area)  
: 사용자의 요구에 따라 생성/소멸

# 1. 동적할당

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  char name[30];
5
6  char* ReadUserName() {
7      printf("What's your name? ");
8      gets(name);
9      return name;
10 }
11
12 int main() {
13     char* name1;
14     char* name2;
15
16     name1 = ReadUserName();
17     printf("name1: %s \n", name1);
18
19     name2 = ReadUserName();
20     printf("name2: %s \n", name2);
21
22     printf("name1: %s \n", name1);
23     printf("name2: %s \n", name2);
24
25     return 0;
26 }
```

CN Microsoft Visual Studio 디버그 콘솔

```
What's your name? Park Sang Hyeon
name1: Park Sang Hyeon
What's your name? Park Jong Hyeon
name2: Park Jong Hyeon
name1: Park Jong Hyeon
name2: Park Jong Hyeon
```

# 1. 동적할당

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  char name[30];
5
6  char* ReadUserName() {
7      printf("What's your name? ");
8      gets(name);
9      return name;
10 }
11
12 int main() {
13     char* name1;
14     char* name2;
15
16     name1 = ReadUserName();
17     printf("name1: %s \n", name1);
18
19     name2 = ReadUserName();
20     printf("name2: %s \n", name2);
21
22     printf("name1: %s \n", name1);
23     printf("name2: %s \n", name2);
24
25     return 0;
26 }
```

CN Microsoft Visual Studio 디버그 콘솔

```
What's your name? Park Sang Hyeon
name1: Park Sang Hyeon
What's your name? Park Jong Hyeon
name2: Park Jong Hyeon
name1: Park Jong Hyeon
name2: Park Jong Hyeon
```

# 1. 동적할당

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int* arr = (int*)malloc(sizeof(int) * 5);
6
7     for (int i = 0; i < 5; i++) {
8         arr[i] = i;
9         printf("%d\n", arr[i]);
10    }
11    free(arr);
12    return 0;
13 }
```

Microsoft

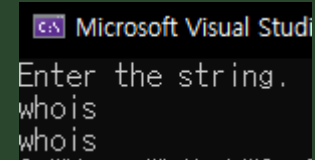
```
0
1
2
3
4
```

(자료형)(포인터명) = (자료형 포인터)malloc(크기(바이트));



# 1. 동적할당

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  int main()
6  {
7      int size;
8
9      printf("Enter the string.\n");
10
11     char* munza = malloc(sizeof(char) * 10);
12
13     scanf("%s", munza);
14     printf("%s", munza);
15
16     free(munza);
17
18     return 0;
19 }
```



Microsoft Visual Studi  
Enter the string.  
whois  
whois

# 1. 동적할당

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  char* ReadUserName() {
5      char* name;
6      name = (char*)malloc(sizeof(char) * 30);
7      printf("What's your name? ");
8      gets(name);
9      return name;
10 }
11
12 int main() {
13     char* name1;
14     char* name2;
15     name1 = ReadUserName();
16     printf("name1: %s \n", name1);
17     name2 = ReadUserName();
18     printf("name2: %s \n\n", name2);
19
20     printf("name1: %s\n", name1);
21     printf("name2: %s\n", name2);
22     free(name1);
23     free(name2);
24     return 0;
25 }
```

C:\> Microsoft Visual Studio 디버그 콘솔

```
What's your name? Park Sang Hyeon
name1: Park Sang Hyeon
What's your name? Park Jong Hyeon
name2: Park Jong Hyeon

name1: Park Sang Hyeon
name2: Park Jong Hyeon
```

## 2. 동적 확장 할당

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int* arr = (int*)malloc(sizeof(int) * 5);
6
7      for (int i = 0; i < 5; i++) {
8          arr[i] = i;
9          printf("%d\n", arr[i]);
10     }
11     printf("\n 이후로 갱신 \n\n");
12     realloc(arr, sizeof(int) * 10);
13
14     for (int j = 0; j < 10; j++) {
15         arr[j] = j;
16         printf("%d\n", arr[j]);
17     }
18
19     free(arr);
20     return 0;
21 }
```

C:\ Microsoft Visu

0  
1  
2  
3  
4  
  
이후로 갱신  
  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9

### 3. 또 다른 할당

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int *arr = (int*)calloc(5, sizeof(int));
6
7      for (int i = 0; i < 5; i++) {
8          arr[i] = i;
9          printf("%d\n", arr[i]);
10     }
11     free(arr);
12     return 0;
13 }
```

Microsoft

```
0
1
2
3
4
```

4. 그 외 +a

# 문자열 잘 다루기

---

1. strlen() 함수
2. strcat(), strncat() 함수
3. strcpy(), strncpy() 함수
4. strcmp(), strncmp() 함수
5. atoi(), atol(), atoll(), atof() 함수
6. toupper(), tolower() 함수

# #1 strlen() : 문자열 길이 조회

---

원형

```
#include <string.h>
size_t strlen(const char *s);
```

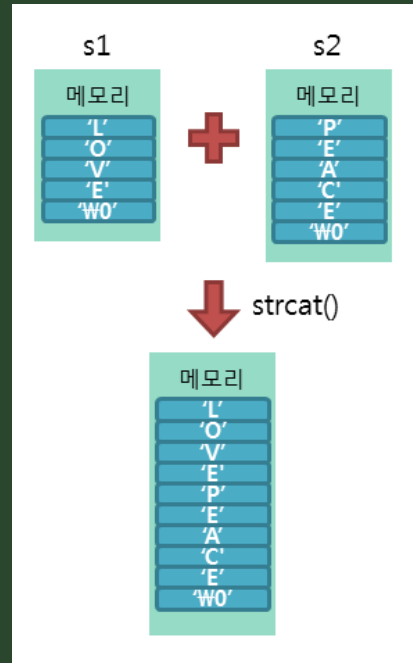
## #2 strcat(), strncat() : 문자열 이어붙이기

원형

```
#include <string.h>
char *strcat(char * restrict s1, const char * restrict s2);
```

원형

```
#include <string.h>
char *strncat(char * restrict s1, const char * restrict s2, size_t n);
```





# #3 strcmp(), strncmp() : 문자열 비교하기

---

원형

```
#include <string.h>
int strcmp(const char *s1, const char *s2);
```

원형

```
#include <string.h>
int strncmp(const char *s1, const char *s2, size_t n);
```