

CS SCHOOL

포인터 사용법 : 2차원 배열 포인터

1. 1차원 배열을 가리키는 배열 포인터

```
int arr[5] = { 1, 2, 3, 4, 5 };  
int * ptr = arr;
```

2차원 배열을 가리키는 포인터는 어떻게 나타낼까요??

2. 2차원 배열을 가리키는 배열 포인터

```
int arr[2][5] =  
{  
    {1, 2, 3, 4, 5},  
    {6, 7, 8, 9, 10}  
};
```

int(*ptr)[5] = arr;
자료형 가로길이

```
int(*ptr)[5] = arr;
```

포인터 사용법 : 2차원 배열 포인터

예제

```
int arr[2][5] =  
{  
    {1, 2, 3, 4, 5},  
    {6, 7, 8, 9, 10}  
};
```

```
int(*ptr)[5] = arr;
```

```
for (int i = 0; i < 2; i++)  
{  
    for (int j = 0; j < 5; j++)  
    {  
        printf("%d    ", ptr[i][j]);  
    }  
    printf("\n");  
}
```

포인터 사용법 : 2차원 배열 포인터

예제

```
int arr[2][5] =  
{  
    {1, 2, 3, 4, 5},  
    {6, 7, 8, 9, 10}  
};
```

```
int(*ptr)[5] = arr;
```

```
for (int i = 0; i < 2; i++)  
{  
    for (int j = 0; j < 5; j++)  
    {  
        printf("%d    ", ptr[i][j]);  
    }  
    printf("\n");  
}
```

typedef의 사용법

배열 포인터를 하다가 뜬금없이 왜 typedef이 나오냐구요?
잠시만 기다려주십시오! 그만한 이유가 있으니까요~

1. typedef의 사용법1 : 기본 사용법

```
typedef unsigned int SIZE_T;
```

지난 시간에 만들었던
STRLEN() 함수입니다.

```
SIZE_T STRLEN(const char * s)
{
    unsigned int i=0;
    while (s[i] != '\0')
    {
        i++;
    }
    return i;
}
```

기존에 있던 기본 데이터형에
우리만의 의미를 부여할 때,
즉 새로운 자료형으로 선언할 때
쓰였습니다.

typedef의 사용법

2. typedef의 사용법2 : 배열 or 배열 포인터

```
typedef int IARR3[3];
typedef double DDARR5[][5];

int main(void)
{
    IARR3 arr1 = { 1, 2, 3 };
    DDARR5 arr2= {
        {1, 2, 3, 4, 5},
        {6, 7, 8, 9, 10},
        {11, 12, 13, 14, 15}
    };
};
```

포인터 사용법

: 2차원 배열 포인터

다시 2차원 배열 포인터로 돌아와서 지금부터 아주 중요한 함수 이야기를 하나 하겠습니다.

1. 함수의 매개변수에 1차원 배열을 넣어야 할 때

```
int * ArrayChange(int arr[], int length, const int num)
{
    for (int i = 0; i < length; i++) arr[i] = num;
    return arr;
}

int main(void)
{
    int arr[5] = { 1, 2, 3, 4, 5 };
    int len = sizeof(arr) / sizeof(int);

    int * ptr = ArrayChange(arr, len, 5);

    for (int i = 0; i < len; i++)
        printf("%d ", ptr[i]);
    puts("");
}
```

방법 1

```
int * ArrayChange(int arr[], int length, const int num)
```

포인터 사용법 : 2차원 배열 포인터

1. 함수의 매개변수에 1차원 배열을 넣어야 할 때

방법 1

```
int * ArrayChange(int arr[], int length, const int num)
```

방법 2

```
int * ArrayChange(int * arr, int length, const int num)
```


포인터 사용법 : 2차원 배열 포인터

2. 함수의 매개변수에 2차원 배열을 넣어야 할 때

방법 1

```
int * ArrayChange(int arr[][5], int length, const int num)
```

방법 2

```
int * ArrayChange(int(*arr)[5], int length, const int num)
```

포인터 사용법 : 2차원 배열 포인터

3. 함수의 반환형에 1차원 배열을 넣어야 할 때

```
int *ArrayChange(int(*arr)[5], int length, const int num)
```

ONLY 한 가지 방법

int []는 인정하지 않습니다!!

4. 함수의 반환형에 2차원 배열을 넣어야 할 때

```
int(*arr)[5] ArrayChange(int(*arr)[5], int length, const int num)
```

(X)

ERROR가 납니다!!!

포인터 사용법 : 2차원 배열 포인터

4. 함수의 반환형에 2차원 배열을 넣어야 할 때

자료형 이름 : 가로 길이가 5인 2차원 배열을 가리키는 포인터

```
typedef int(*twoDimArrPtr)[5];
```

```
twoDimArrPtr ArrayChange(int(*arr)[5], int length, const int num)
```

자료형이므로 반환 가능

Study 과제 1

: 1차원 배열 2개를 매개변수로 받아서 2차원 배열을 만들어
반환하는 함수를 정의하시오. (인터페이스는 아래 참조)
(만약 열이 5보다 크면 5열까지만 작다면 나머지 부분을 0으로 채우시오.)

```
typedef int(*twoDimArrPtr)[5];
```

```
twoDimArrPtr MergeTwoArr(int ar1[], int * ar2, int lenAr1, int lenAr2)  
                                ar1의 길이      ar2의 길이
```

포인터 사용법 : 함수 포인터

1. 함수 이름과 함수포인터 변수

함수이름은 함수 위치의 주소값

```
void Function(void) {}
```

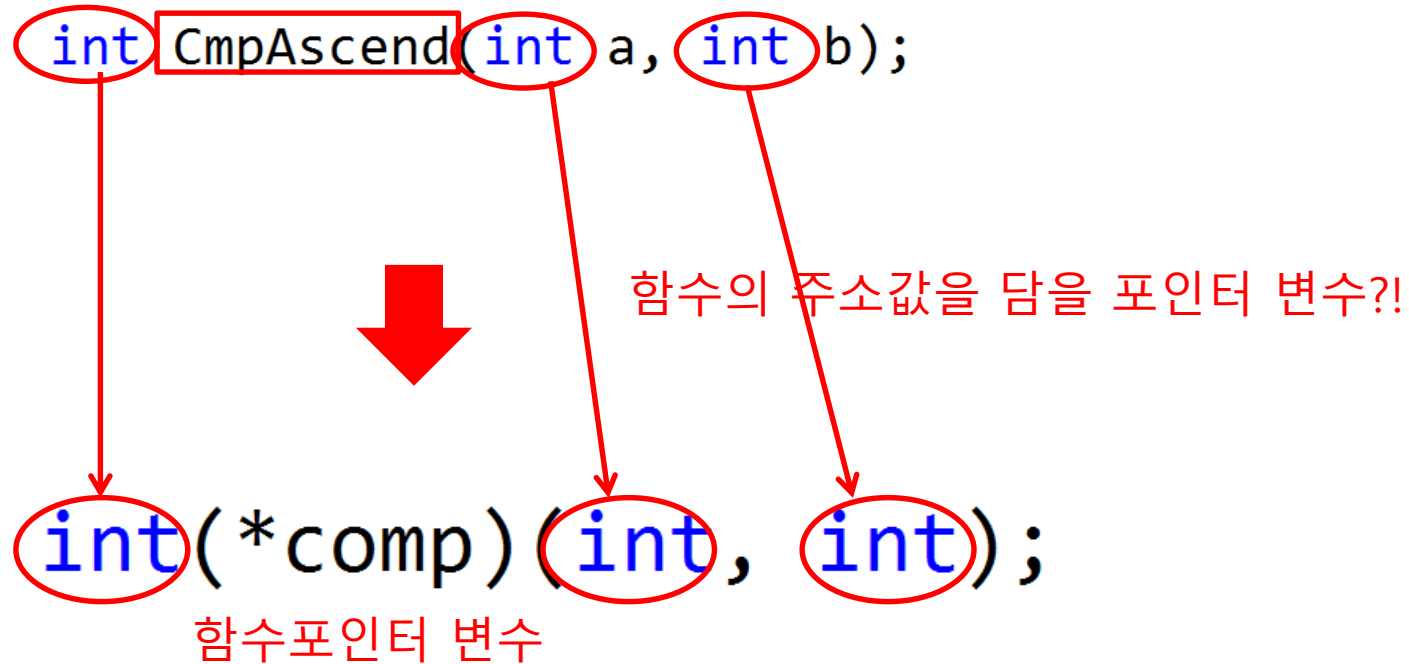
```
int main(void)
{
    printf("%#p \n", Function);

    return 0;
}
```

함수포인터는 C/C++에서 활용 범위가 굉장히 넓습니다!!!
반드시 마스터하셔야 해요!!

포인터 사용법 : 함수 포인터

1. 함수 이름과 함수포인터 변수



포인터 사용법 : 함수 포인터

2. 함수포인터로 함수 호출하기

```
int AddTwo(int a, int b)
{
    return a + b;
}
```

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

함수 포인터 문법에 익숙해집시다!

```
int(*ptr)(int, int) = AddTwo;  
int num1 = 10, num2 = 20;
```

마치 함수처럼 호출합니다!!

```
int num3 = ptr(num1, num2);  
printf("num3 : %d \n", num3);
```

포인터 사용법 : 함수 포인터

3. 함수포인터를 매개변수로

```
int AddTwo(int a, int b)
{
    return a + b;
}
```

매개변수는 형식 그대로 넣어주시면 됩니다.

```
int CallAddFunc(int(*funcPtr)(int, int), const int n1, const int n2)
{
    funcPtr(n1, n2);
}
```

```
int main(void)
{
    int value = CallAddFunc(AddTwo, 10, 20);
    printf("value : %d \n", value);
}
```

포인터 사용법 : 함수 포인터

4. 함수포인터를 반환형으로

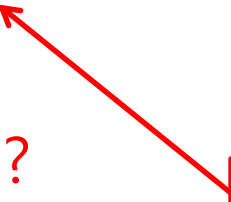
반환형

`[int(*funcPtr)(int, int)]`

어떻게 표현해야 할까?

```
int AddTwo(int a, int b)
{
    return a + b;
}

[ ] GetFunc(void)
{
    return AddTwo;
}
```



포인터 사용법 : 함수 포인터

4. 함수포인터를 반환형으로


이것은 문법입니다. 외우는 수 밖에 없어요 πππ

```
int AddTwo(int a, int b)
{
    return a + b;
}
```

```
int AddTwo(int a, int b)
{
    return a + b;
}
```

```
GetFunc(void)
{
    return AddTwo;
}
```

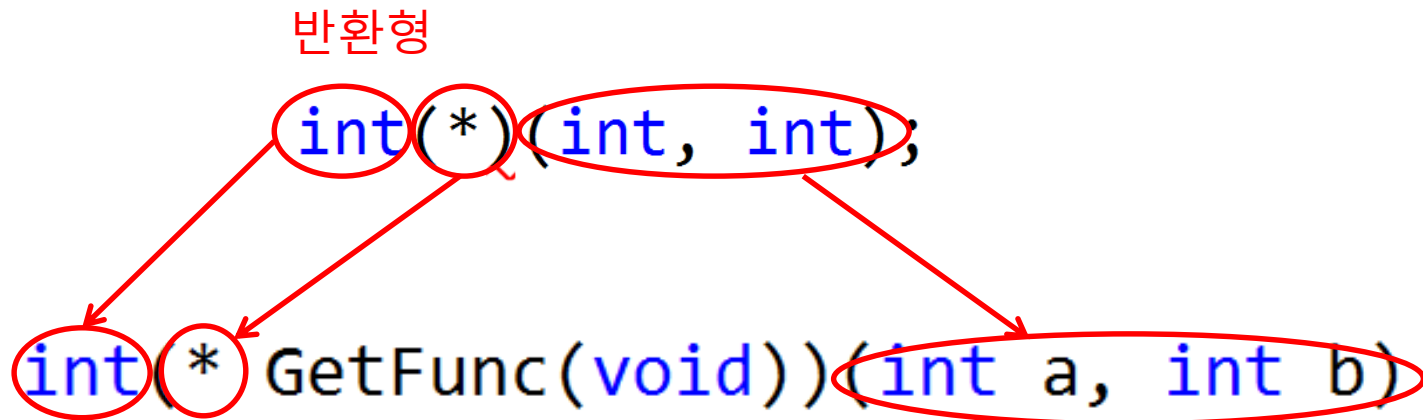
```
int(* GetFunc(void))(int a, int b)
{
    return AddTwo;
}
```



포인터 사용법 : 함수 포인터

4. 함수포인터를 반환형으로

이것은 문법입니다. 외우는 수 밖에 없어요 $\pi\pi$



포인터 사용법 : 함수 포인터

4. 함수포인터를 반환형으로

```
int AddTwo(int a, int b)
{
    return a + b;
}
```

```
int(* GetFunc(void))(int a, int b)
{
    return AddTwo;
}
```

```
int main(void)
{
    int(*ptr)(int, int) = GetFunc();
    int value = ptr(10, 20);
    printf("value : %d \n", value);
}
```

포인터 사용법
: 함수 포인터

5. 함수포인터를 반환형으로
: typedef을 사용해 보다 쉽게 바꾸기

```
int AddTwo(int a, int b)
{
    return a + b;
}
```

typedef이 붙는 순간 AddFunc는 새로운 자료형이 됩니다.

```
typedef int(*AddFunc)(int, int);
```

```
AddFunc GetFunc(void)
{
    return AddTwo;
}
```

포인터 사용법 : 함수 포인터

6. 함수포인터 예제

```
1  #include <stdio.h>
2
3  int CmpAscend(int a, int b);
4  int CmpDescend(int a, int b);
5
6  typedef int(*comp)(int, int);
7
8  int * ArrangeArr(int arr[], comp func, int lenArr);
9
10 int main(void)
11 {
12     int arr[] = { 3, 1, 2, 4, 10, 8, 5, 7, 6, 9 };
13     int len = sizeof(arr) / sizeof(int);
14     int button;
```

예제는 뒤에 계속 됩니다!

포인터 사용법 : 함수 포인터

6. 함수포인터 예제

```
15
16     do
17     {
18         printf("정렬방법을 선택하세요.(오름차순 :1, 내림차순 :2) : ");
19         scanf("%d", &button);
20     } while (button != 1 && button != 2);
21
22     switch (button)
23     {
24     case 1:
25         ArrangeArr(arr, CmpAscend, len);
26         break;
27     case 2:
28         ArrangeArr(arr, CmpDescend, len);
29         break;
30     default:
31         printf("잘못 입력하셨습니다.");
32     }
```

예제는 뒤에 계속 됩니다!

함수포인터의 쓰임

포인터 사용법 : 함수 포인터

6. 함수포인터 예제

예제는 뒤에 계속 됩니다!

```
33  
34     for (int i = 0; i < len; i++)  
35         printf("%d ", arr[i]);  
36  
37     return 0;  
38 }
```

```
39  
40 int CmpAscend(int a, int b)  
41 {  
42     if (a > b)  
43         return 1;  
44     else  
45         return -1;  
46 }
```

오름차순 함수

두 함수의 인터페이스가 같다는 걸
기억하세요! → 함수포인터를 쓰기 위해!

```
47  
48 int CmpDescend(int a, int b)  
49 {  
50     if (a < b)  
51         return 1;  
52     else  
53         return -1;  
54 }
```

내림차순 함수

포인터 사용법 : 함수 포인터

6. 함수포인터 예제

```
56 int * ArrangeArr(int arr[], comp func, int lenArr)
57 {
58     for (int i = 0; i < lenArr - 1; i++)
59     {
60         for (int j = 0; j < (lenArr - i) - 1; j++)
61         {
62             if (func(arr[j], arr[j + 1])>0)
63             {
64                 int temp = arr[j];
65                 arr[j] = arr[j + 1];
66                 arr[j + 1] = temp;
67             }
68         }
69     }
70
71     return arr;
72 }
```

버블정렬 핵심 코드