

# C Programmin Mentoring

---

## 포인터

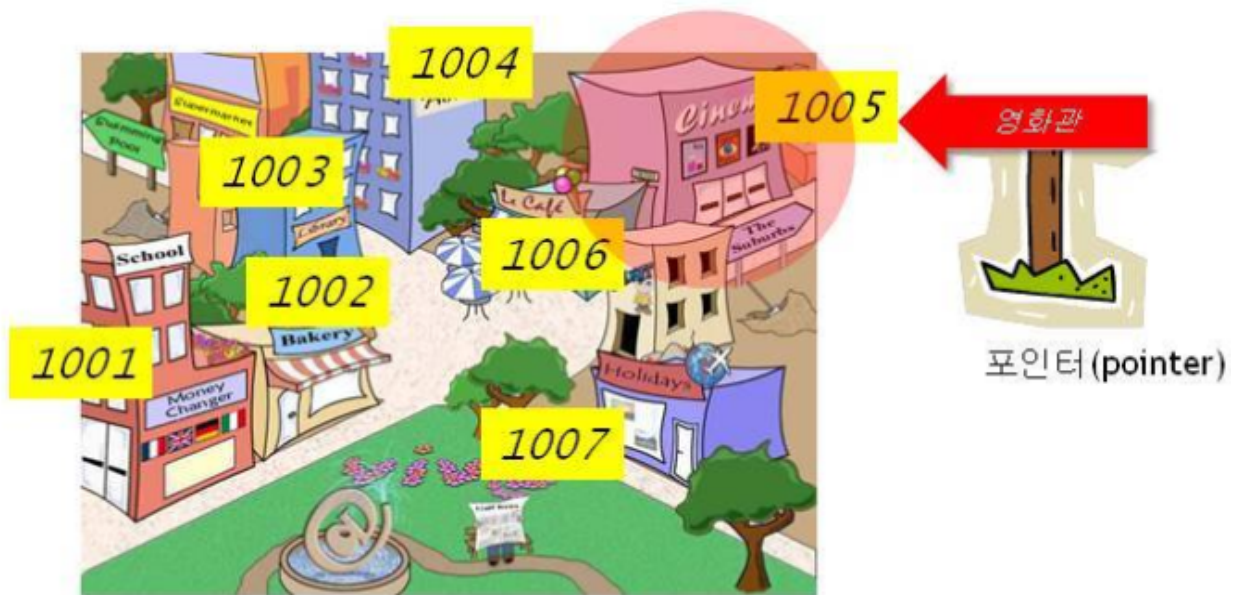
---

### 이론

---

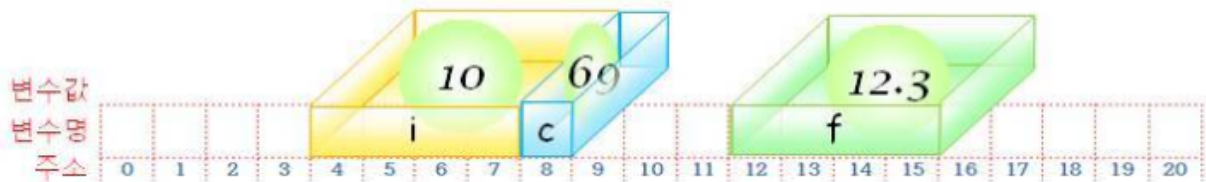
#### 포인터란?

- **포인터(pointer):** 주소를 가지고 있는 변수



- 변수의 크기에 따라서 차지하는 메모리 공간이 다름
- char 형: 1바이트; int 형: 4바이트; ...

```
int i = 10;
char c = 69;
float f = 12.3F;
```



## 변수의 주소

```
int main(void)
{
    int i = 10;
    char c = 69;
    float f = 12.3F;

    printf("i의 주소: %u\n", (unsigned)&i); // 변수 i의 주소 출력
    printf("c의 주소: %u\n", (unsigned)&c); // 변수 c의 주소 출력
    printf("f의 주소: %u\n", (unsigned)&f); // 변수 f의 주소 출력
    return 0;
}
```

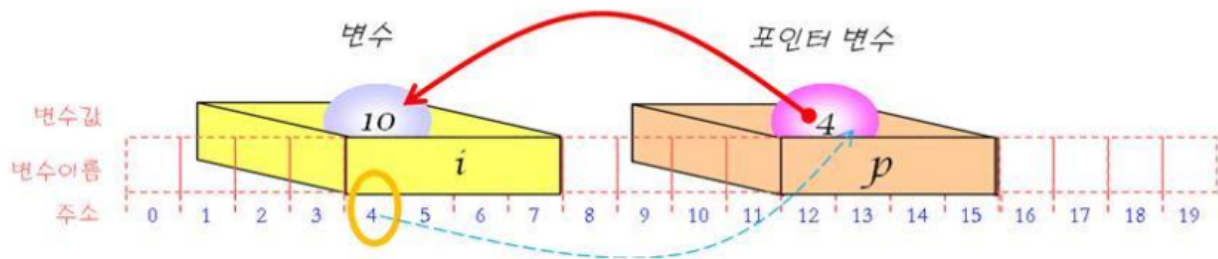
```
i의 주소: 1245024
c의 주소: 1245015
f의 주소: 1245000
```

## 간접 참조 연산자 \*

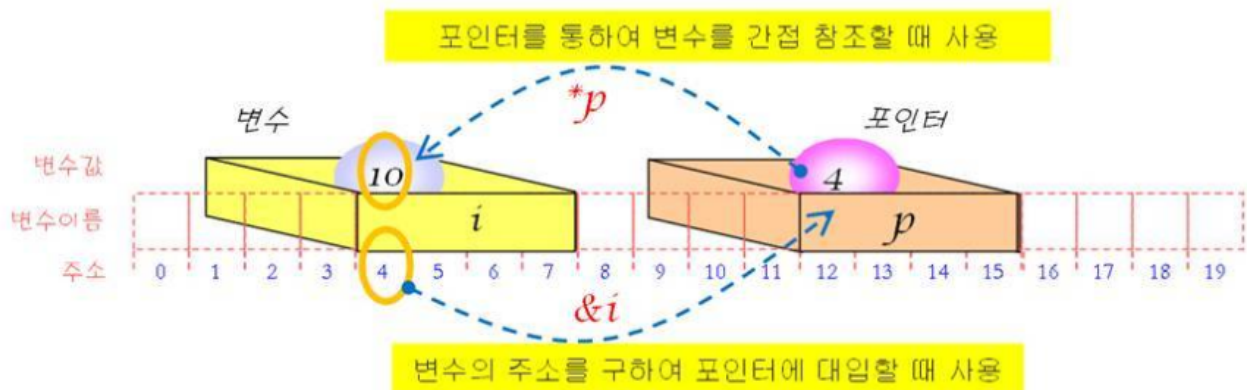
간접 참조란 포인터가 가리키는 주소에 저장된 값을 가져오는 것이다.

### 포인터 변수

- `int i = 10;`
- `int *p;` // 정수 포인터 변수 선언  
`p = &i;` // p는 i를 가리킴
- `int *p = &i;` // 정수 포인터 변수 선언 및 초기화



### & 연산자와 \* 연산자



# 포인터 예제

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

```
int main(void)
```

```
{
```

```
    int i = 3000;
```

```
    int *p = &i;           // 변수와 포인터 연결
```

```
    printf("i = %d\n", i);    // 변수의 값 출력
```

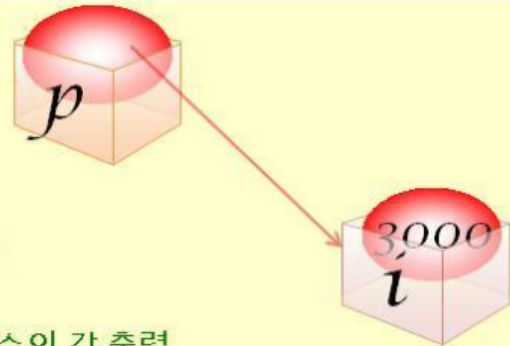
```
    printf("&i = %u\n", (unsigned)&i); // 변수의 주소 출력
```

```
    printf("**p = %d\n", *p); // 포인터를 통한 간접 참조 값 출력
```

```
    printf("p = %u\n", (unsigned)p); // 포인터의 값 출력
```

```
    return 0;
```

```
}
```



```
i = 3000
&i = 1245024
*p = 3000
p = 1245024
```

# 포인터 예제

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

```
int main(void)
```

```
{
```

```
    int x = 10, y = 20;
```

```
    int *p;
```

```
    p = &x;
```

```
    printf("p = %u\n", (unsigned)p);
```

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

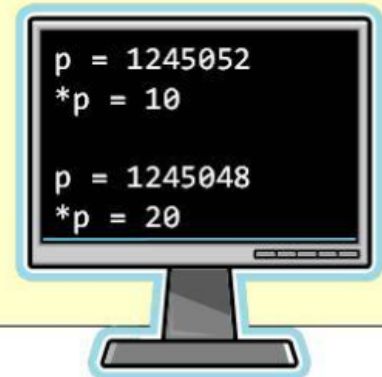
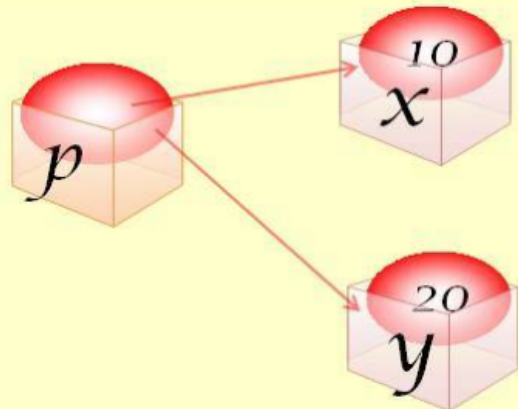
```
    p = &y;
```

```
    printf("p = %u\n", (unsigned)p);
```

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

```
    return 0;
```

```
}
```



## • 중간점검

1. 메모리는 어떤 단위를 기준으로 주소가 매겨지는가?
2. 다음의 각 자료형이 차지하는 메모리 공간의 크기를 써라.  
(a) char (b) short (c) int (d) long (e) float (f) double
3. 포인터도 변수인가?
4. 변수의 주소를 추출하는데 사용되는 연산자는 무엇인가?
5. 변수 x의 주소를 추출하여 변수 p에 대입하는 문장을 써라.
6. 정수형 포인터 p가 가리키는 위치에 25를 저장하는 문장을 써라.

## 포인터 사용시 주의점

- 초기화되지 않은 포인터를 사용하면 안 됨

```
int *p;           // p는 초기화 되어 있지 않음
*p = 100;         // 위험!
```

- NULL** 포인터: 아무것도 가리키고 있지 않는 포인터
  - `#define NULL 0`
  - 아무것도 가리키고 있지 않을 경우, 포인터를 NULL로 설정
  - `p = NULL;`  
`p = 0;`
  - `if (p == NULL) ...` // p가 아무것도 가리키지 않으면  
`if (p == 0) ...`  
`if (!p) ...`
  - `if (p != NULL) ...` // p가 무엇이든 가리키고 있으면  
`if (p != 0) ...`  
`if (p) ...`

- 중간점검

- 초기값이 결정되지 않은 포인터에는 어떤 값을 넣어두는 것이 안전한가?
-



## 포인터 연산

### 간접 참조 연산자와 증감 연산자

- $*p++ \rightarrow *(p++)$ : p를 나중에 증가
- $(*p)++$ : \*p를 나중에 증가, p가 가리키는 값 증가
- $v = *p++;$      $\rightarrow$   $v = *p; p++;$     // p를 나중에 증가
- $v = (*p)++;$      $\rightarrow$   $v = *p; (*p)++;$     // \*p를 나중에 증가
- $v = *++p;$      $\rightarrow$   $++p; v = *p;$     // p를 먼저 증가
- $v = ++*p;$      $\rightarrow$   $++(*p); v = *p;$     // \*p를 먼저 증가

## 포인터 연산

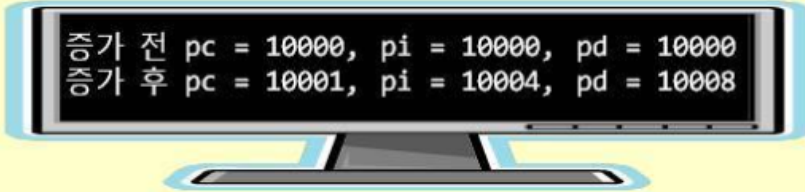
```
#include <stdio.h>

int main(void)
{
    char *pc; int *pi; double *pd;

    pc = (char*)10000; pi = (int*)10000; pd = (double*)10000;
    printf("증가 전 pc = %d, pi = %d, pd = %d\n", (int)pc, (int)pi, (int)pd);

    pc++; pi++; pd++;
    printf("증가 후 pc = %d, pi = %d, pd = %d\n", (int)pc, (int)pi, (int)pd);

    return 0;
}
```



```
증가 전 pc = 10000, pi = 10000, pd = 10000
증가 후 pc = 10001, pi = 10004, pd = 10008
```

- 중간점검
  1. int형 포인터 p가 80번지를 가리키고 있었다면 (p+1)은 몇 번지를 가리키고 있는가?
  2. p가 포인터라고 하면  $*p++$ 와  $(*p)++$ 의 차이점은 무엇인가?
  3. p가 char형 포인터이고 10번지라고 하면  $*(p+3)$ 의 의미는 무엇이고 몇 번지를 가리키고 있는가?

## 포인터와 배열

## 포인터와 배열

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

```
int main() {
```

```
    int a[] = { 10, 20, 30, 40, 50 };
```

```
    printf("&a[0] = %u\n", (unsigned)&a[0]);
```

```
    printf("&a[1] = %u\n", (unsigned)&a[1]);
```

```
    printf("&a[2] = %u\n", (unsigned)&a[2]);
```

```
    printf("a = %u\n", (unsigned)a);
```

```
    return 0;
```

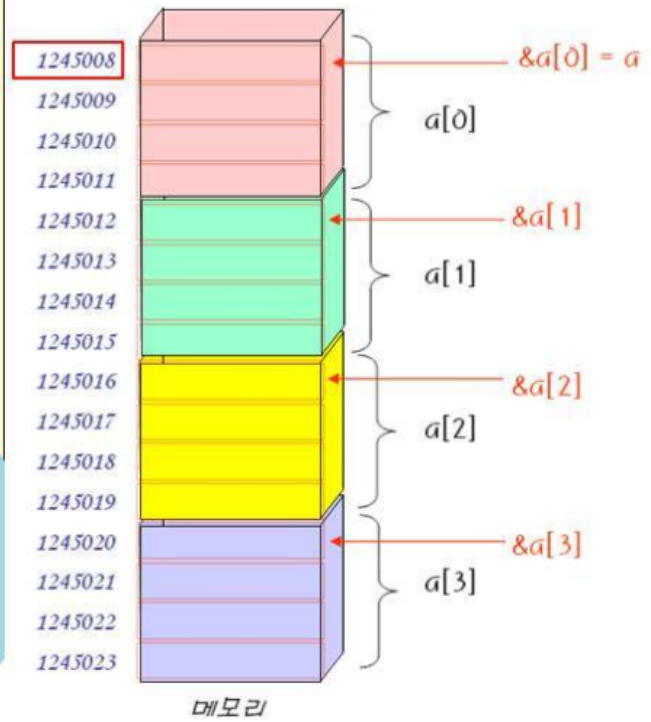
```
}
```

```
&a[0] = 1245008
```

```
&a[1] = 1245012
```

```
&a[2] = 1245016
```

```
a = 1245008
```





## 배열 역순 출력

```
#include <stdio.h>
void print_reverse(const int a[ ], int n);

int main( ) {
    int a[ ] = { 10, 20, 30, 40, 50 };
    print_reverse(a, sizeof a / sizeof a[0]);
    return 0;
}

void print_reverse(const int a[ ], int n) {
    const int *p = a + n - 1;    // 마지막 원소를 가리키도록 초기화
    while (p >= a)
        printf("%d\n", *p--);    // *(p--)
}
```



- 중간점검

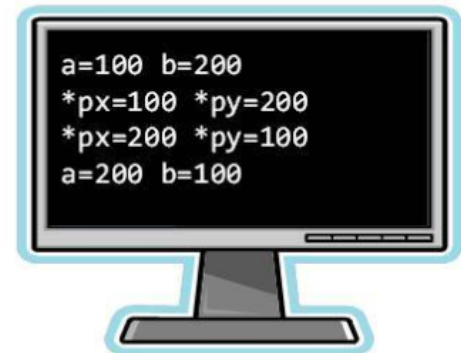
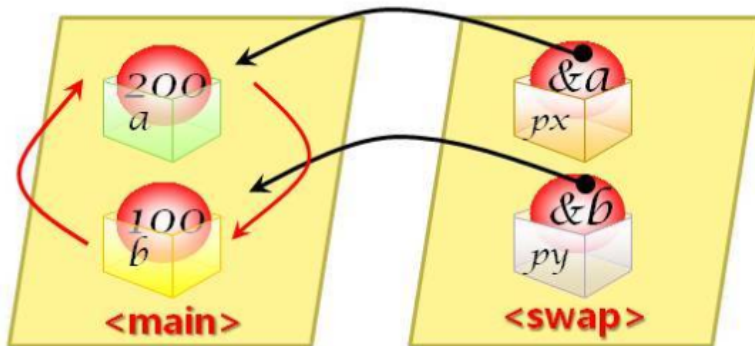
1. 배열의 첫 번째 원소의 주소를 계산하는 2가지 방법은 ?
2. 배열 a[ ] 에서 \*a의 의미는 무엇인가?
3. 배열의 이름에 다른 변수의 주소를 대입할 수 있는가?
4. 포인터를 이용하여 배열의 원소들을 참조할 수 있는가?
5. 포인터를 배열의 이름처럼 사용할 수 있는가?

## 포인터와 함수

### swap 함수

```
#include <stdio.h>
void swap(int *px, int *py);
int main() {
    int a = 100, b = 200;
    printf("a=%d b=%d\n", a, b);
    swap(&a, &b);
    printf("a=%d b=%d\n", a, b);
    return 0;
}
```

```
void swap(int *px, int *py) {
    int tmp;
    printf("**px=%d *py=%d\n", *px, *py);
    tmp = *px; *px = *py; *py = tmp;
    printf("**px=%d *py=%d\n", *px, *py);
}
```



- 중간점검

1. 함수에 매개 변수로 변수의 복사본이 전달되는 것을 라고 한다.
2. 함수에 매개 변수로 변수의 원본이 전달되는 것을 라고 한다.
3. 배열을 함수의 매개 변수로 지정하는 경우, 배열의 복사가 일어나는가?

## 실습

- 2개의 정수의 합과 차를 동시에 반환하는 함수를 작성하고 테스트하라. 포인터 매개 변수를 사용한다.

```
#include <stdio.h>

void get_sum_diff(int x, int y, int *p_sum, int *p_diff);

int main() {
    int x, y;
    int sum, diff;

    printf("2개의 정수를 입력하십시오: ");
    scanf("%d %d", &x, &y);

    get_sum_diff(x, y, &sum, &diff);

    printf("x + y = %d \n", sum);
    printf("x - y = %d \n", diff);

    return 0;
}

void get_sum_diff(int x, int y, int *p_sum, int *p_diff) {
    *p_sum = x + y;
    *p_diff = x - y;
}
```

- 정수 배열을 받아서 원소들을 난수로 채우는 함수를 작성하고 테스트하라. 난수는 라이브러리 함수인 rand()를 사용하여 생성한다.

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

#define SIZE 10

void array_fill(int *A, int size);

int main() {
    int arr[SIZE];
    int i;
    srand((unsigned)time(NULL));

    array_fill(arr, SIZE);
    for (i = 0; i < SIZE; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}

void array_fill(int *A, int size) {
```

```
int i;  
for (i = 0; i<size; i++)  
    *(A+i) = 1 + rand() % 100;  
}
```