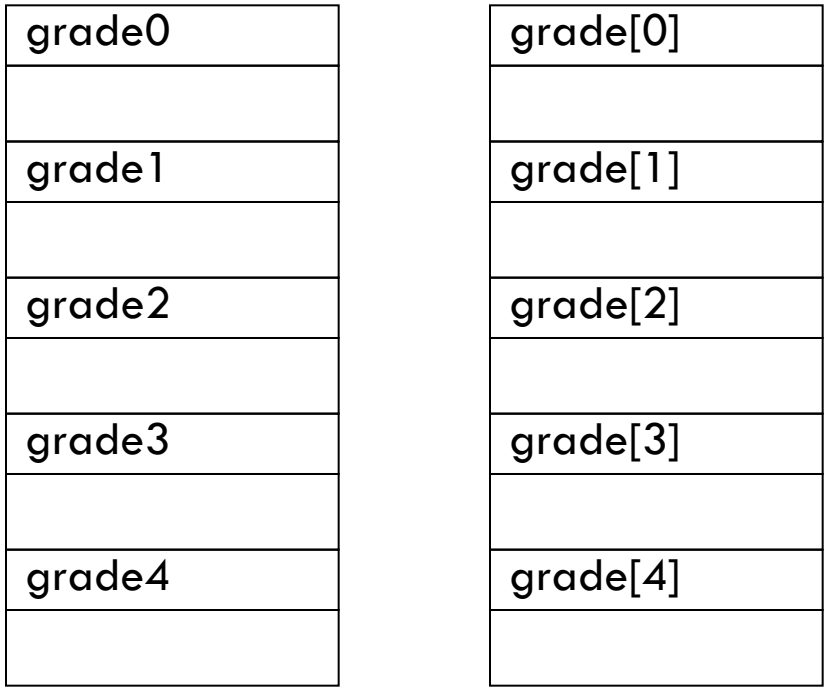
c 기초 특강

배열, 포인터

1차원 배열

- 비교

```
int grade0, grade1, grade2, grade3, grade4;
int grade[5];
```



배열 선언과 첨자

- 선언 형식

  type array_name[*size*];

- 선언 예

  int grade[5];

- 첨자(index) : 문장에서 쓰이는 대괄호 안의 수
- 첨자 사용 예

  grade[2] = 10;

- 첨자의 범위
  - 배열의 크기가 *size*인 경우 첨자는 0부터 *size* – 1까지

배열의 초기화(1)

- 전통적인 C : 외부와 정적 배열만 배열 초기자 이용 가능
- ANSI C : 자동 배열도 초기화 가능
- 사용 예

```
double    real[5] = {0., 1., 2., 3., 4.};
```

- 초기화 목록의 개수가 배열 원소 개수보다 작은 경우 나머지는 모두 0으로 초기화

```
int          a[100] = {0};
```

배열의 초기화(2)

- 외부와 정적 배열은 명시적으로 초기화하지 않아도 디폴트로 모두 0으로 초기화
- 자동 배열은 반드시 초기화를 시켜야
- 배열의 크기를 생략한 경우 초기자의 개수가 크기가 된다

```
int a[] = {2, 3, 4, -7};        /* a[]의 크기는 4 */
```

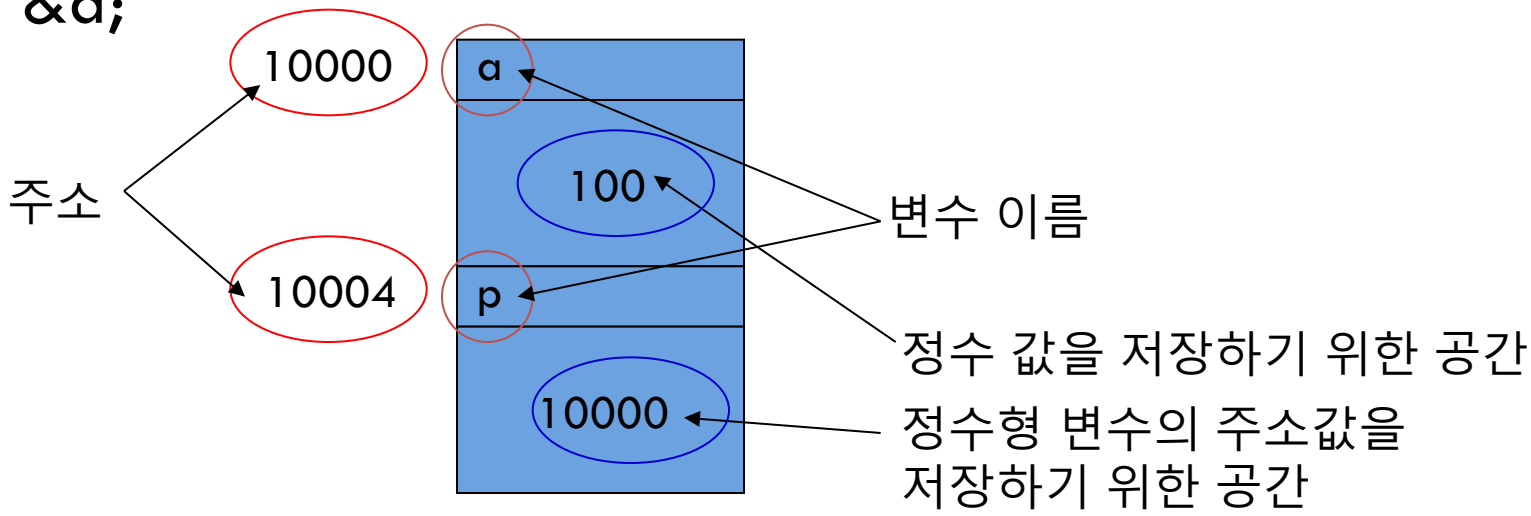
- 문자열의 경우 마지막의 \0 유의

```
char s[] = "abc";               /* 다음과 동일 */
char s[4] = {'a','b','c','\0'};
```

포인터

- 변수의 3 요소 : 이름, 주소, 공간
- 포인터 : 주소 값을 저장하기 위한 변수

```
int a, *p;
a = 100;
p = &a;
```



& 연산자와 * 연산자

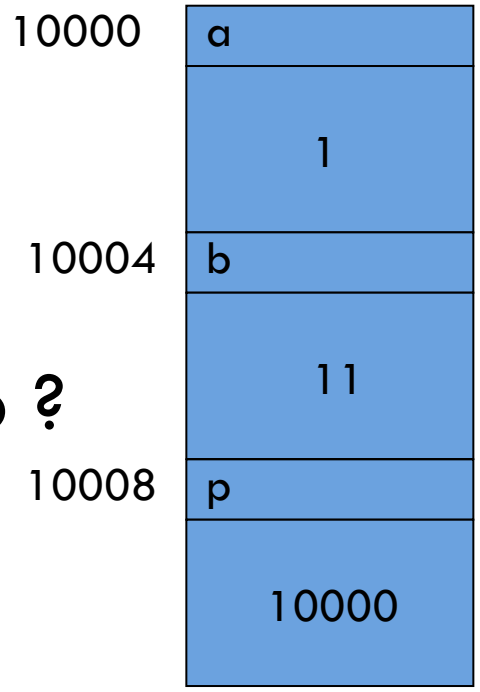
- & 연산자 : 객체의 주소
- * 연산자 : 저장되어 있는 주소로 객체를 찾아가도록 하는 연산자

```
int a = 1, b = 2, *p;

p = &a;

b = *p + 10;
```

- [Quiz] 그림에서 &a, &b, &p, *p ?

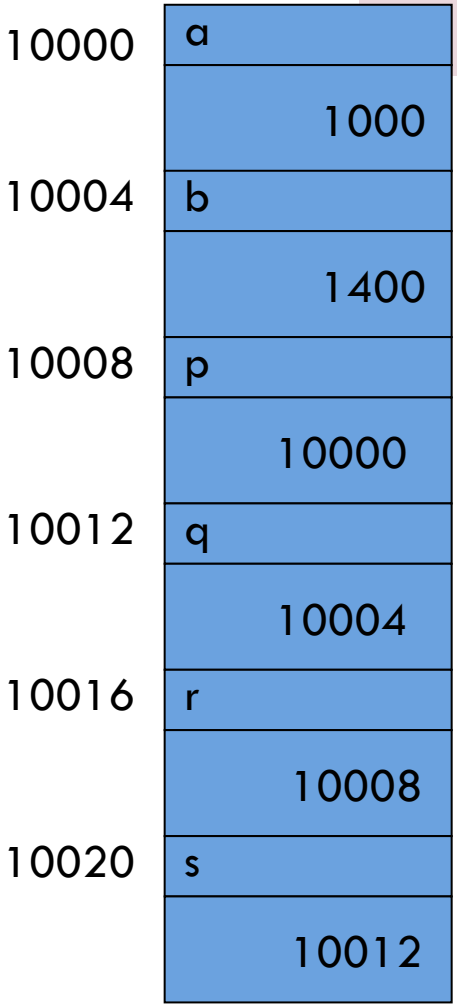


& 연산자와 * 연산자 예제

```
int       a, b, *p, *q, **r, **s;

a  = 100;
p = &a;
q = &b;
*q = 300;
printf("address of a : %p\t value of a : %d\n", &a, a);
printf("address of b : %p\t value of b : %d\n", &b, b);
printf("pointer operator - : p - q = %d \n", p - q);
printf("int operator - : p - q = %d \n", (int)p - (int)q);

r = &p;
s = &q;
**r = 1000;
**s = *p + 400;
printf("address of r : %p\t value of r : %d\n", &r, r);
printf("address of s : %p\t value of s : %d\n", &s, s);
```



주소 값 출력 예

- %p 형식

```
/* Printing an address, or location. */
#include   <stdio.h>
int main(void)
{
    int   i = 7, *p = &i;

    printf("%s%d\n%s%p\n",
           "     Value of i: ", *p,
           "  Location of i: ", p);
    return 0;
}
```

참조에 의한 호출

- 매개변수에서 변수 이름을 쓰면 변수에 저장된 값이 복사되어 전달
- 매개변수로 쓰이는 변수가 함수의 지역 변수가 되므로 호출된 함수의 값 변화는 함수 내부에서만 유효
- 함수를 호출한 함수에서도 값이 변하도록 하기 위해서는 반드시 **주소가 전달**되어야 한다

참조에 의한 호출 예제

- swap()에서 주소를 전달 받았기에 main()안의 i와 j 값이 수정될 수 있다.

```
void swap(int *p, int *q)
{
    int tmp;
    tmp = *p;
    *p = *q;
    *q = tmp;
}
```

```
void swap(int *, int *);
int main(void)
{
    int i = 3, j = 5;
    swap(&i, &j);
    printf("%d %d\n", i, j);
        /* 5  3 is printed */
    return 0;
}
```

배열과 포인터의 관계(1)

- 배열의 이름은 **고정된** 주소 또는 포인터 **값**
- 배열과 포인터 모두 첨자 이용 가능
- C에서는 배열을 포인터를 이용 구현

```
int  a[100], *p;

a[0] = 100;
a[10] = 400;

p = a;
printf("address of p = %p, value of p = %p, value of *p = %d\n", &p, p, *p);
printf("address of a[0] = %p, value of a[0]= %d\n", &a[0], a[0]);

printf("value of *(p+10) = %d, value of a[10] = %d\n", *(p+10), a[10]);
printf("value of p[10] = %d, value of *(a+10) = %d\n", p[10], *(a+10));
```

배열과 포인터의 관계(2)

```
#define   N           100
int         * p, a[N], sum ;
```

Version 1

```
    for (i = 0, sum = 0; i < N; ++i)
        sum += a[i] ;          /* 또는  sum += *(a + i) ; */
```

Version 2

```
    for (p = a, sum = 0; p < &a[N]; ++p)
        sum += *p ;
```

Version 3

```
    for (p = a, i = 0, sum = 0; i < N; ++i)
        sum += p[i] ;
```

함수 인자로서의 배열

- 다음 두 선언은 동일

```
double sum(double a[], int n)      /* n is the size a[] */
{
        .....


double sum(double *a, int n)     /* n is the size a[] */
{
        .....
```

과제

- 2차방정식 프로그램을 함수를 이용하여 작성
  - 사용자에게 3개의 수를 받는 함수
  - 2차방정식 처리하는 함수
  - 2차방정식의 해를 출력하는 함수
  - 앞의 함수들을 호출하는 main()
- hms2sec 프로그램
  1. 사용자에게 시, 분, 초를 받아 초단위 합을 출력하는 프로그램을 작성
  2. 시, 분, 초를 매개변수로 받아 초를 반환하는 함수 작성
  3. 2에서 작성한 함수를 main()호출하여 확인하는 프로그램 작성
  4. 사용자 입력과 출력 부분을 보완하여 최종 프로그램 작성
- sec2hms 프로그램
  - 사용자에게 초를 입력받아 시, 분, 초를 출력하는 프로그램 작성