Java를 알고 C배우기

컴퓨터프로그래밍3

week 6-3 포인터와 배열-포인터와 문자열

2022.1학기

충남대 조은선

배열 이름과 포인터 관계 대부분 복습

아래와 같은 프로그램에 대해,

```
int arr[10];
int * ptr;
ptr = arr;
```

- 배열 이름(arr)과 포인터(ptr)는 거의 동일
  `arr[2], *(arr + 2), ptr[2], *(ptr + 2)` 4가지 모두 사용 가능

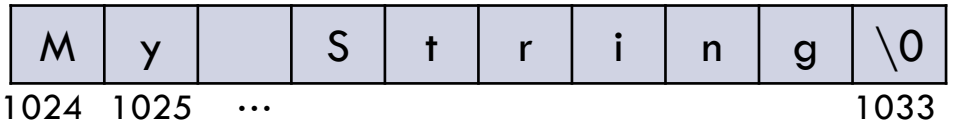
- 그러나 차이점
  배열 이름(arr)은 지정 연산 왼쪽에 올 수 없음 (즉, 값이 불변)
  `ptr = arr;` 은 가능하나, `arr = ptr;` 은 불가

문자열의 표현

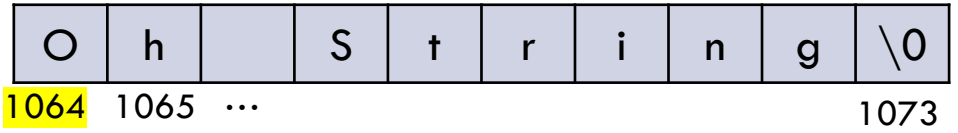
- 문자열 정의 방법 두가지

```
char str1[] = "My String";
    // (1) 배열 초기화 방법 중 하나
```

str1




```
char * str2  = "Oh String";
    // (2) 문자열을 자동 할당
    //      이 문자열은 변화 불가
```



1064

char* 형 포인터변수 str2 (값이 1064)

자동 할당된 문자열 (불변)

```
int main() {
        char * s1 = "your team";
        s1 = "Our team";                    // OK
        s1[2] = 'a';                        // XX (불변!)
        …
}

int main() {
        char s2[] = "its team";
        s2 = "Their team";                  // XX (불변!)
        s2[2] = 'a';                        //  OK
}
```

- s1[2]의 'u'를 'a'로 바꿔 치기 할 수 없음
  이유: 자동 할당 된 불변 문자열
- s2에 "Their team"을 지정할 수 없음
  이유: 배열 이름이므로 (앞페이지!)

자동 할당 문자열이란?

char 배열 선언할 때 초기화 값으로 쓰이는 경우 말고
그 외에 모든, 큰 따옴표로 둘러싸인 문자열

```
char * str = "Const String";
```

문자열 저장 후 주소 값 반환

```
char * str = 0x1234;
```

문자열이 먼저 할당된 이후에
그 때 반환되는 주소 값이 저장되는 방식이다.

```
printf("Show your string");
```

문자열 저장 후 주소 값 반환

```
printf(0x1234);
```

위와 동일하다.
문자열은 선언 된 위치로 주소 값이 반환된다.

포인터의 배열

- 배열인데, 요소의 타입이 포인터인 배열

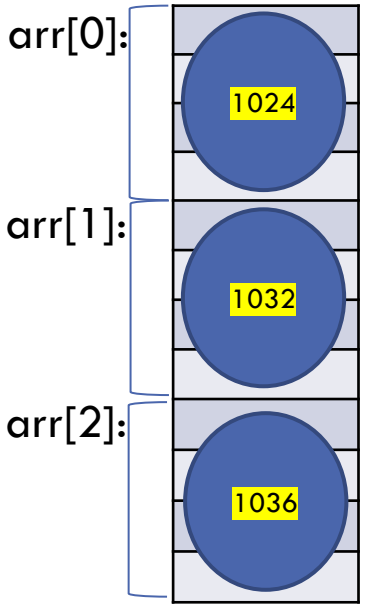
```
int    arr[10];   // 요소의 타입이 int
double darr[10];  // 요소의 타입이 double
int *  arr1[10];  // 요소의 타입이 * (즉, 주소!)
```

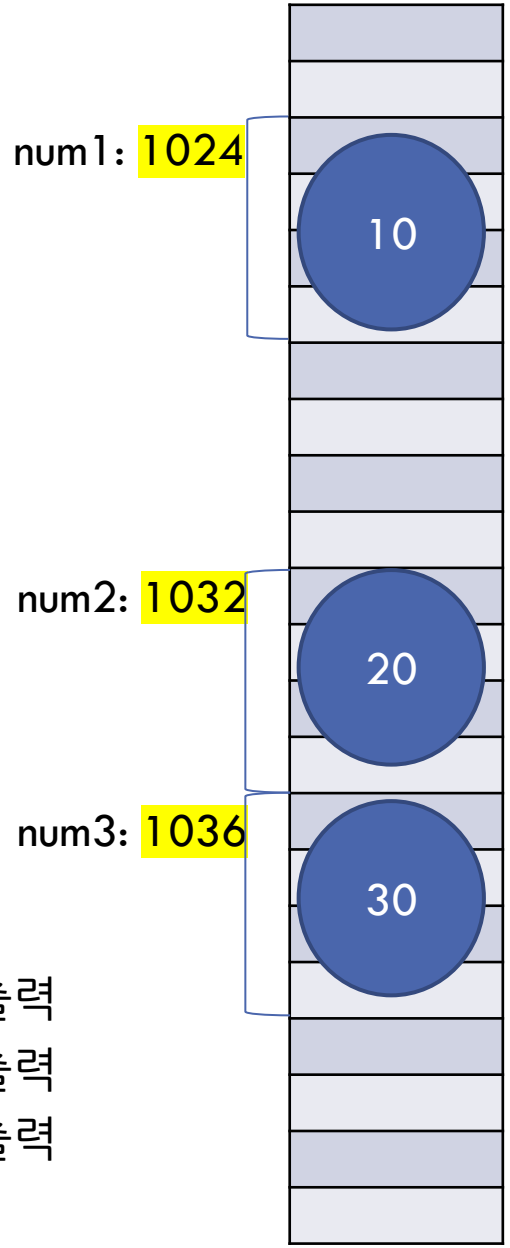
- 예)

```
int num1=10; int num2=20, num3=30;
int * arr[3] = {&num1, &num2, &num3};
// 각 요소의 타입이 int * 인, 3칸 짜리 배열 선언

printf("%d ", *arr[0]);   // arr[0]에 저장된 주소로 가서 값을 읽어다가 출력
printf("%d ", *arr[1]);   // arr[1]에 저장된 주소로 가서 값을 읽어다가 출력
printf("%d ", *arr[2]);   // arr[2]에 저장된 주소로 가서 값을 읽어다가 출력

    // 주소니까 앞에다 *연산자를 붙임
    // 실행 결과 10 20 30
```

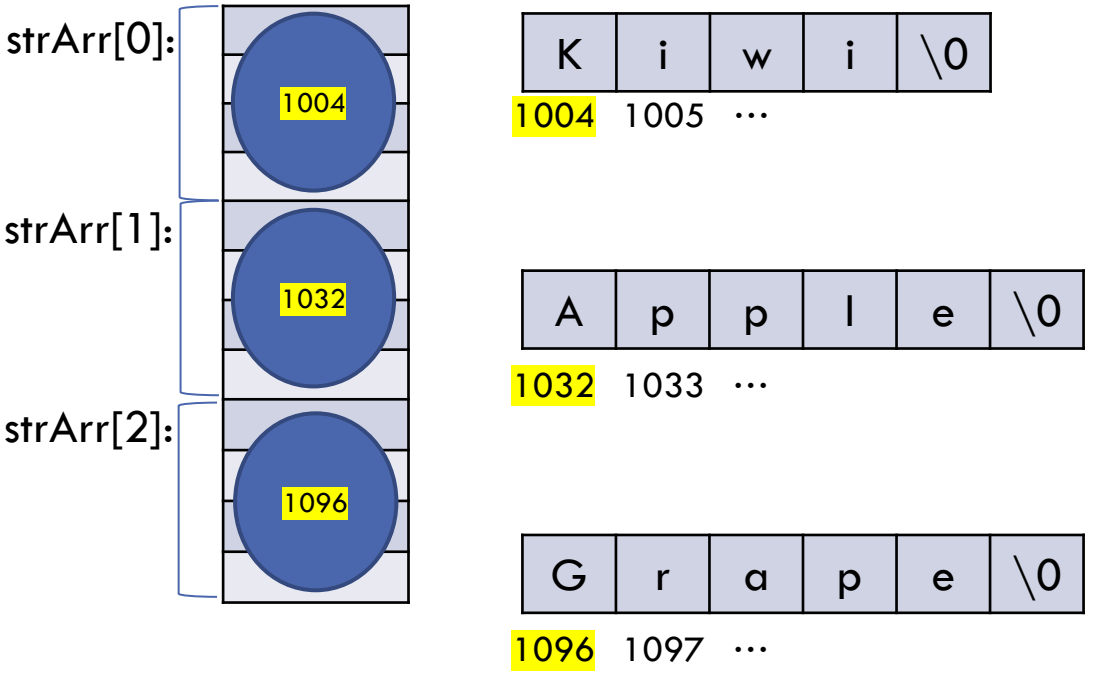




문자열을 위한 char* 타입 포인터의 배열

- 일반 포인터의 배열과 다르지 않다

```
char * strArr[3] = {“Kiwi”, “Apple”, “Grape”};
printf(“%s ”, strArr[0]);
printf(“%s ”, strArr[1]);
printf(“%s ”, strArr[2]);
// 실행결과 Kiwi Apple Grape
```



- 내부적으로는 이렇게 변환됨

```
char * strArr[3] = {“Kiwi”, “Apple”, “Grape”};
```

```
char * strArr[3] = {1004, 1032, 1096};
```

Quiz

- 다음 코드의 출력을 적으시오.

```
char a[]   = “Hello”;
char * p[] = {a, a+1, a+2, a+3, a+4};
       // = {&a[0], &a[1], &a[2], &a[3], &a[4]}
       // ~ {“Hello”, “ello”, “llo”, “lo”, “o”}

printf(“%p %s %c %s %c \n”);
printf(“%s %c \n”, a, *a);
printf(“%s %c \n”, a+1, *(a+1)); // &a[1], a[1]
printf(“%p %p %s \n”, p, *p, *p);
printf(“%s %s %s %s\n”, *p, *(p+1),
                        *(p+2), *(p+3));
```