Chapter 5
Pointers & Arrays
포인터 + 배열

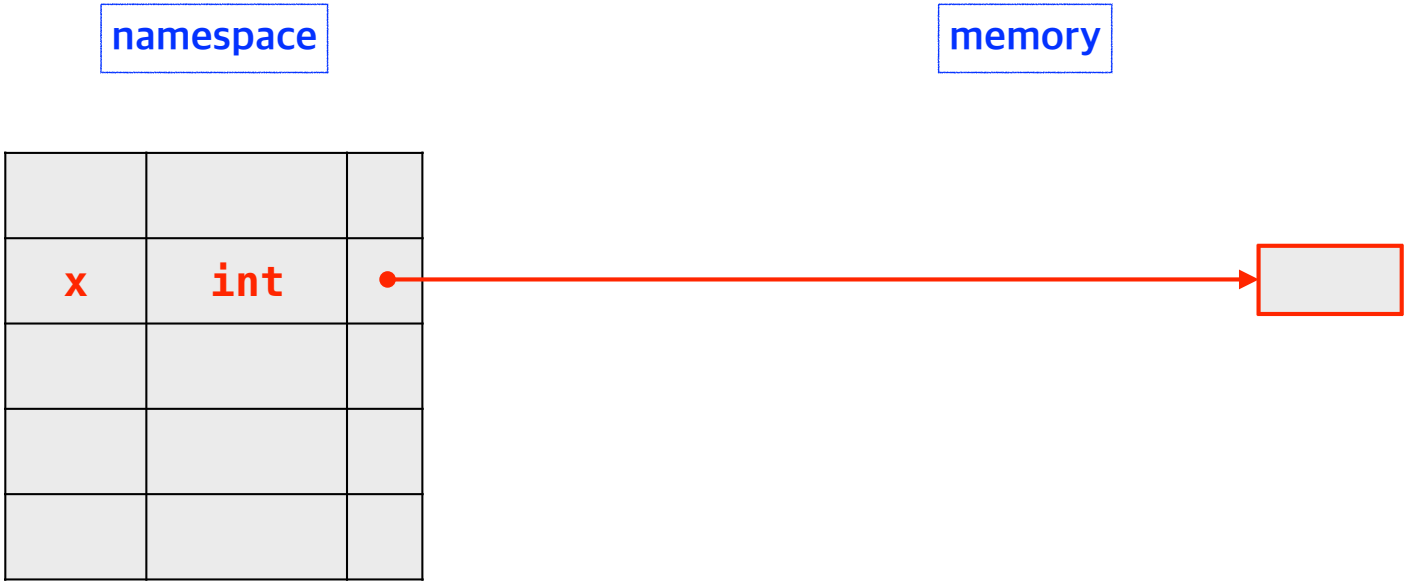
Part I
5.1~5.4

CSE2018 시스템프로그래밍기초
2016년 2학기

한양대학교 ERICA
컴퓨터공학과 => 소프트웨어학부
도경구

1. Pointers & Addresses
2. Pointers & Function Arguments
3. Pointers & Arrays
4. Address Arithmetic
5. Character Pointers & Functions
6. Pointer Arrays; Pointers to Pointers
7. Multi-dimensional Arrays
8. Initialization of Pointer Arrays
9. Pointers vs. Multidimensional Arrays
10. Command-line Arguments
11. Pointers to Functions
12. Complicated Declarations

```
int x;

x = 1;
x = x + 2;
```

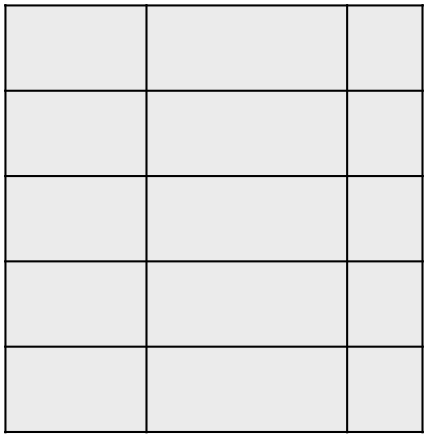


namespace

memory

```
int x;

x = 1;
x = x + 2;
```


```
int x;
int *px;

x = 1;
px = &x;
*px = *px + 2;
```



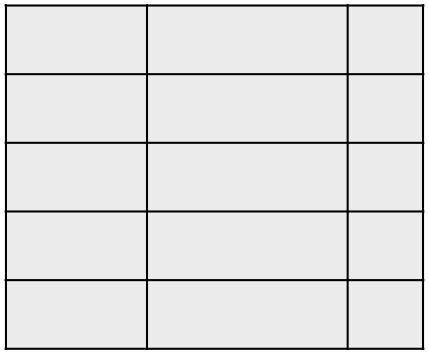
포인터(pointer) **변수**

변수의 주소를 저장하는 변수

```c
int x = 1, y = 2, z[10];
int *ip;        /* ip is a pointer to int */

ip = &x;        /* ip now points to x */
y = *ip;        /* y is now 1 */
*ip = 0;        /* x is now 0 */
ip = &z[0];     /* ip now points to z[0] */
```

namespace

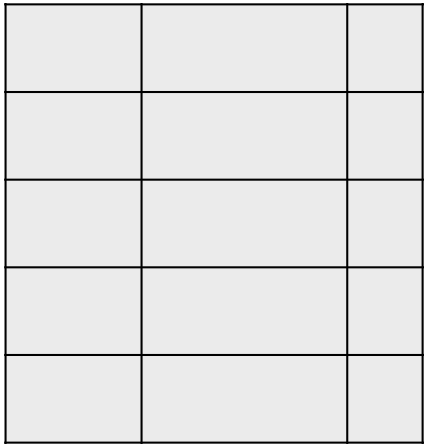
memory



```
int main() {
    int a = 3, b = 7;

    swap(a, b);
}
```

```
void swap(int x, int y) {
    int temp;

    temp = x;
    x = y;
    y = temp;
}
```

namespace



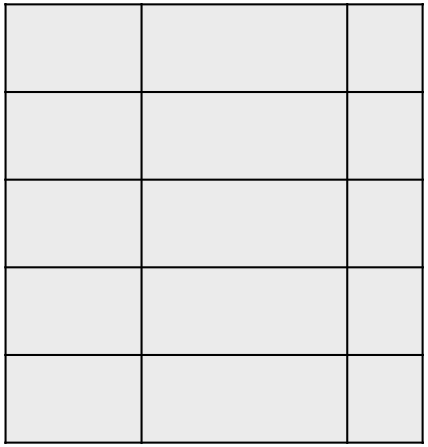
memory

```
int main() {
    int a = 3, b = 7;

    swap(&a, &b);
}
```

```
void swap(int *px, int *py) {
    int temp;

    temp = *px;
    *px = *py;
    *py = temp;
}
```

namespace

memory



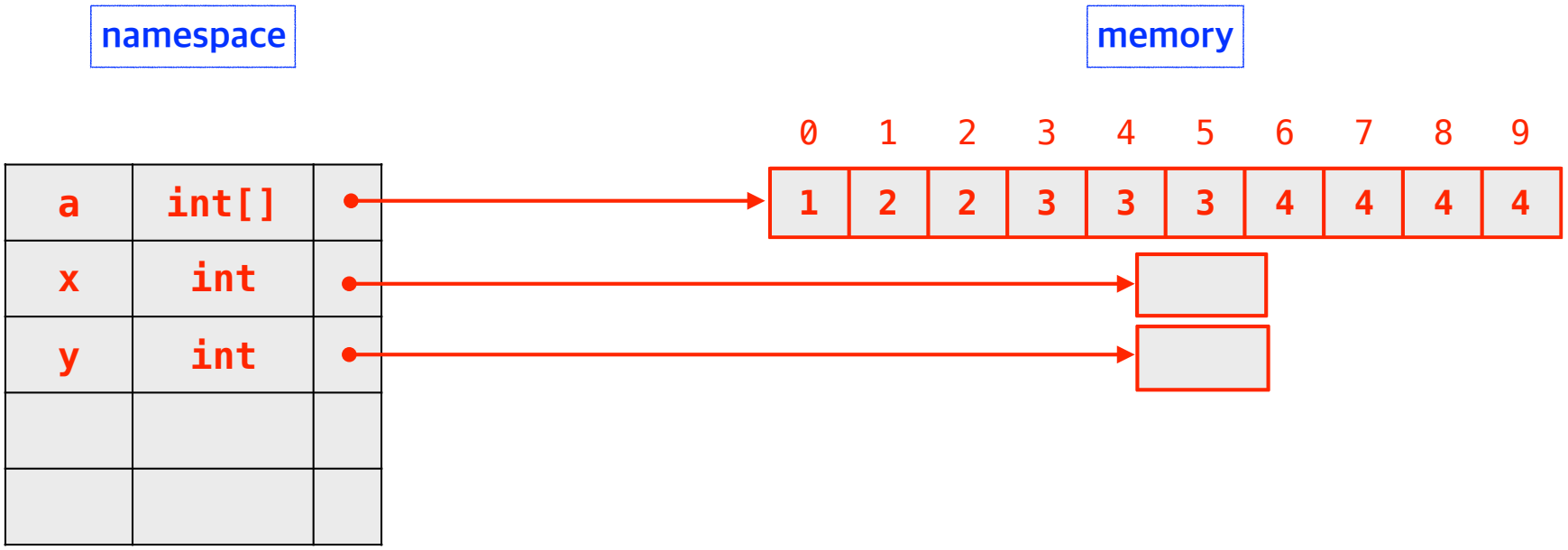
정수 읽어오기 getint

getint.c

```
int main() {
    int n, array[SIZE], getint(int *);

    for (n = 0; n < SIZE && getint(&array[n]) != EOF; n++)
        printf("%d\n", array[n]);
}
```

```
/* getint: get next integer from input into *pn */
int getint(int *pn) {
    int c, sign;

    while (isspace(c = getch()))
        ;
    if (!isdigit(c) && c != EOF && c != '+' && c != '-') {
        ungetch(c);
        return 0;
    }
    sign = (c == '-') ? -1 : 1;
    if (c == '+' || c == '-')
        c = getch();
    for (*pn = 0; isdigit(c); c = getch())
        *pn = 10 * *pn + (c - '0');
    *pn *= sign;
    if (c != EOF)
        ungetch(c);
    return c;
}
```

```c
int a[10];
int x, y;
int *pa;        /* pa is a pointer to int */

pa = &a[0];     /* pa = a; */
x = *pa;        /* x = a[0]; */
y = *(pa+3);    /* y = a[3]; */
pa = a+3;       /* pa = &a[3] */
```



Array vs. Pointer

```
int a[];
int *pa;
```

a[i] ↔ *(a+i)
&a[i] ↔ a+i

pa[i] ↔ *(pa+i)
&pa[i] ↔ pa+i

a = pa ⛔
a++ ⛔

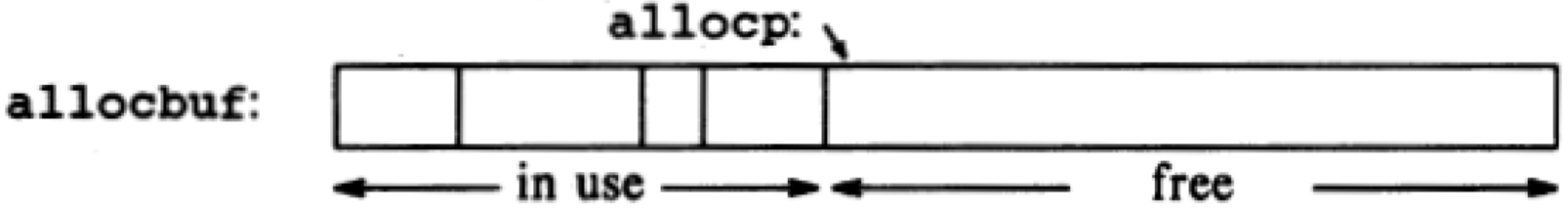
pa = a ✅
pa++ ✅

```c
/* strlen: return length of string s */
int strlen(char *s) {  /* same as char s[] */
    int n;

    for (n = 0; *s != '\0'; s++)
        n++;
    return n;
}
```

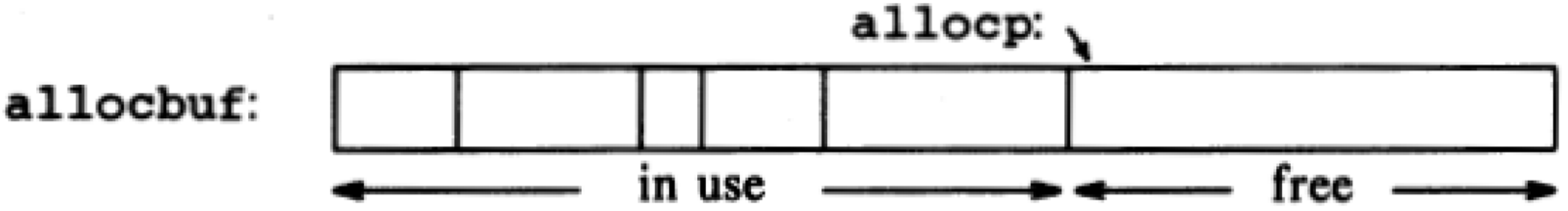
Rudimentary Storage Allocator

```
#define ALLOCSIZE 10000 /* size of available space */

static char allocbuf[ALLOCSIZE]; /* storage for alloc */
static char *allocp = allocbuf; /* next free position */
/* static char *allocp = &allocbuf[0]; */
```

before call to **alloc**:



after call to **alloc**:



Rudimentary Storage Allocator

```
/* alloc: return pointer to n characters */
char *alloc(int n) {
    if (allocbuf + ALLOCSIZE - allocp >= n) { /* it fits */
        allocp += n;
        return allocp - n; /* old p */
    }
    else /* not enough room */
        return NULL; /* return 0; */
}

/* afree: free storage pointed to by p */
void afree(char *p) {
    if (p >= allocbuf && p < allocbuf + ALLOCSIZE)
        allocp = p;
}
```

```c
/* strlen: return length of string s */
int strlen(char *s) {
    int n;

    for (n = 0; *s != '\0'; s++)
        n++;
    return n;
}
```

```c
/* strlen: return length of string s */
int strlen(char *s) {
    char *p = s;

    while (*p != '\0')
        p++;
    return p - s;
}
```

Pointer Arithmetic
(Address Arithmetic)

가능한 포인터 연산

- 같은 타입의 포인터변수에 지정assignment
- 포인터와 정수의 덧셈/뺄셈
- 동일 배열 원소끼리 포인터 뺄셈/크기비교
- 0으로 지정 또는 0과의 비교

이외의 연산은 모두 불가능

- ~~포인터끼리의 덧셈, 곱셈, 나눗셈, …~~
- ~~포인터와 실수의 덧셈~~
- ~~다른 타입의 포인터변수에 지정~~