Object oriented programming In C++ (ACE 1004)

Pointer

Professor 최학남

xncui@inha.ac.kr

Office: high-tech 401

Contents

- Definition of **pointer**
- Address of variable
- Declaration of **pointer**
- **Pointer** operation

Definition of pointer

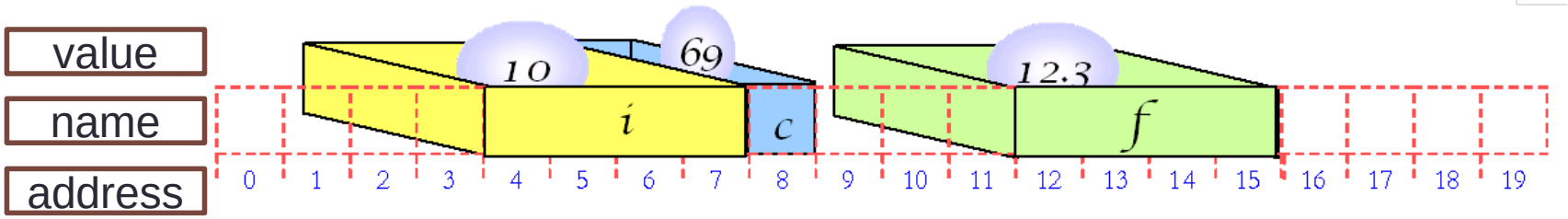
- Pointer is a memory address of a variable
  - ✓Example
    - int n = 0;
    - int *num
      - Create a pointer num
    - num = &n;
      - & operator returns the address of its argument
      - Set the pointer num to the address of the variable n

- x, &x, *x
  - ✓x : the value of x
  - ✓&x : the address of x
  - ✓*x : another variable pointed to by x.
    - x should contain an address, and *x is the data in that memory address.

Variable and memory

- *char* variable : 1 byte
- *int* type variable : 4 byte
- *float* type variable : 4 byte
- *double* type variable : 8 byte

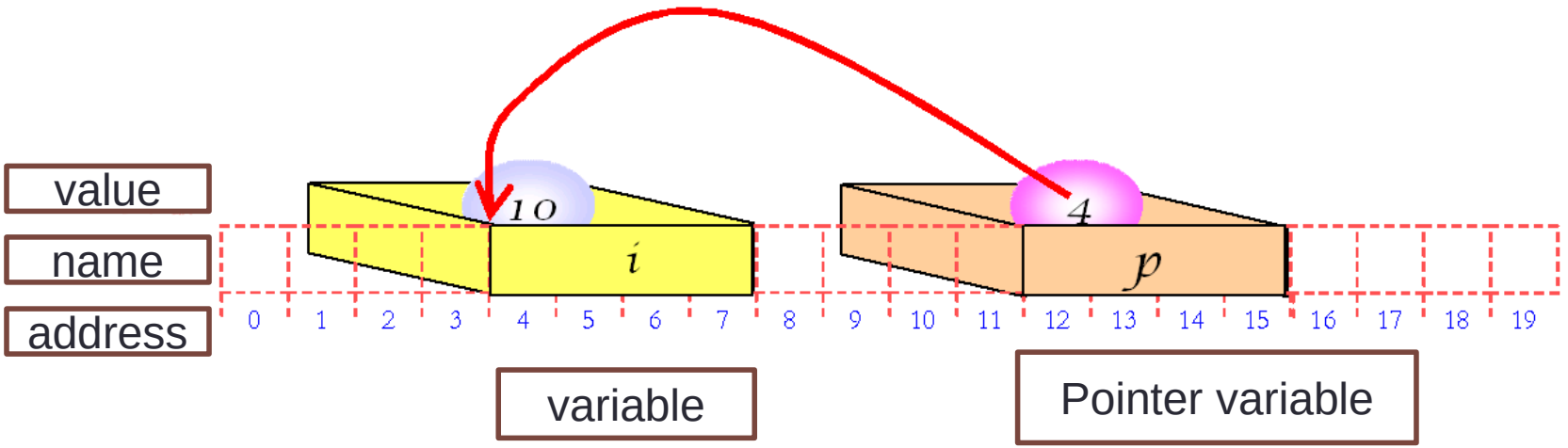
```
void main(void)
{
  int i = 10;
  char c = 69;
  float f = 12.3;
}
```



Declaration of pointer

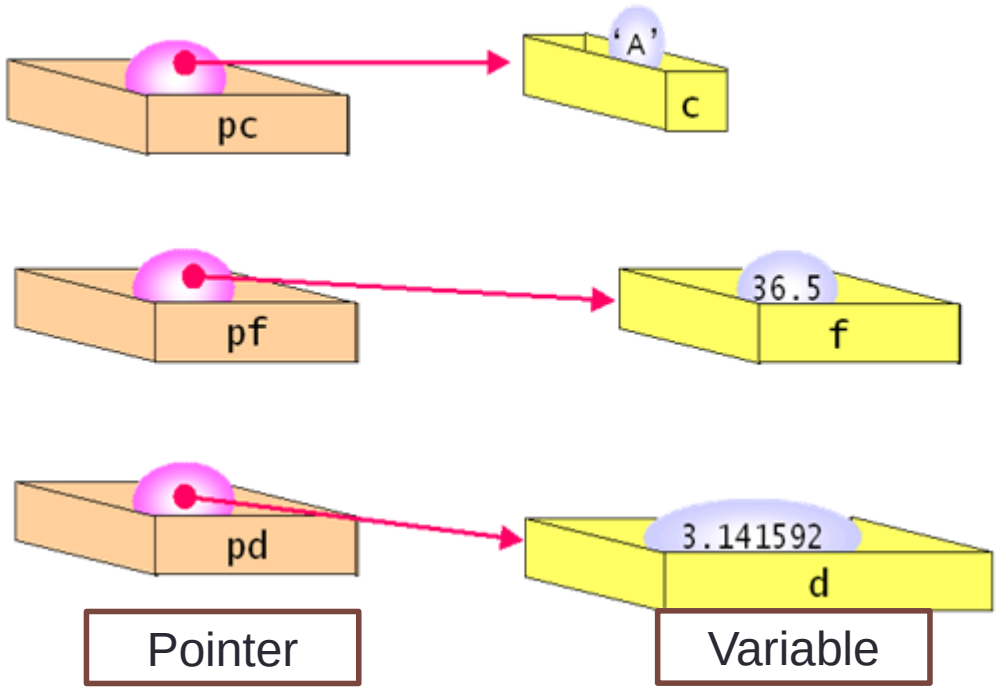
- Pointers are variables which point to memory locations

```
int i = 10;          // declaration of integer data type i
int *p = &i;         // address of variable i assigned to the pointer p
```



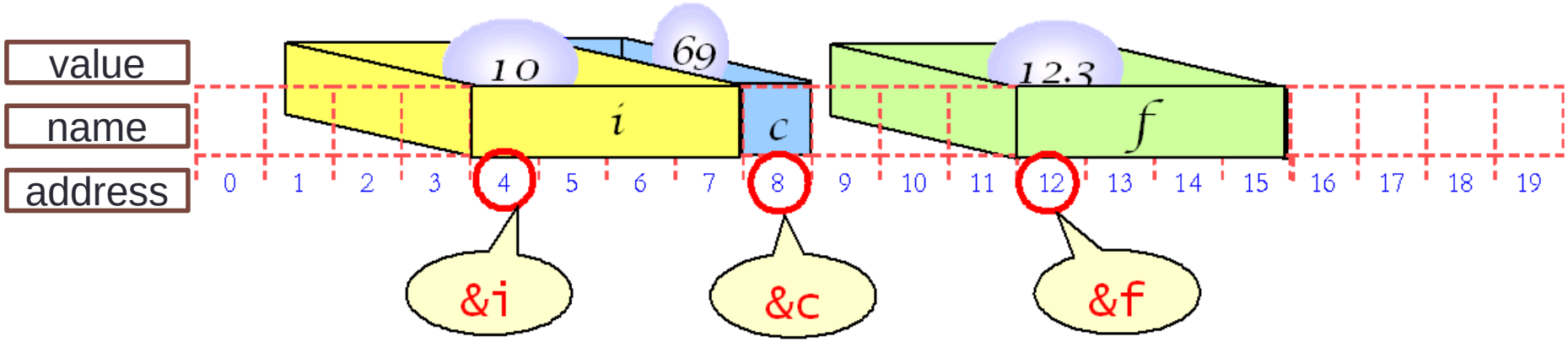
Declaration of pointer

```
char c = 'A';                  // char type variable  c
float f = 36.5;                // float type variable f
double d = 3.141592;           // double type variable d

char *pc = &c;                 // pc point to the char c
float *pf = &f;                // pf point to the float f
double *pd = &d;               // pd point to the double d
```



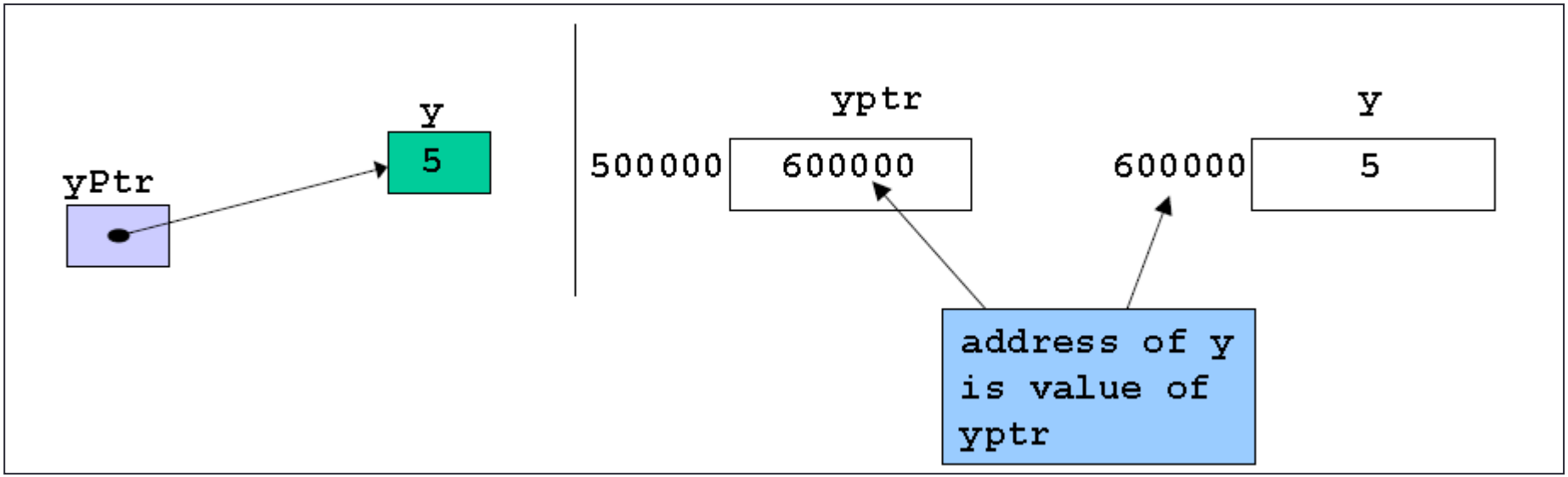
Pointer operators

- Address operator: &
- Address of variable i: &i



Pointer operators

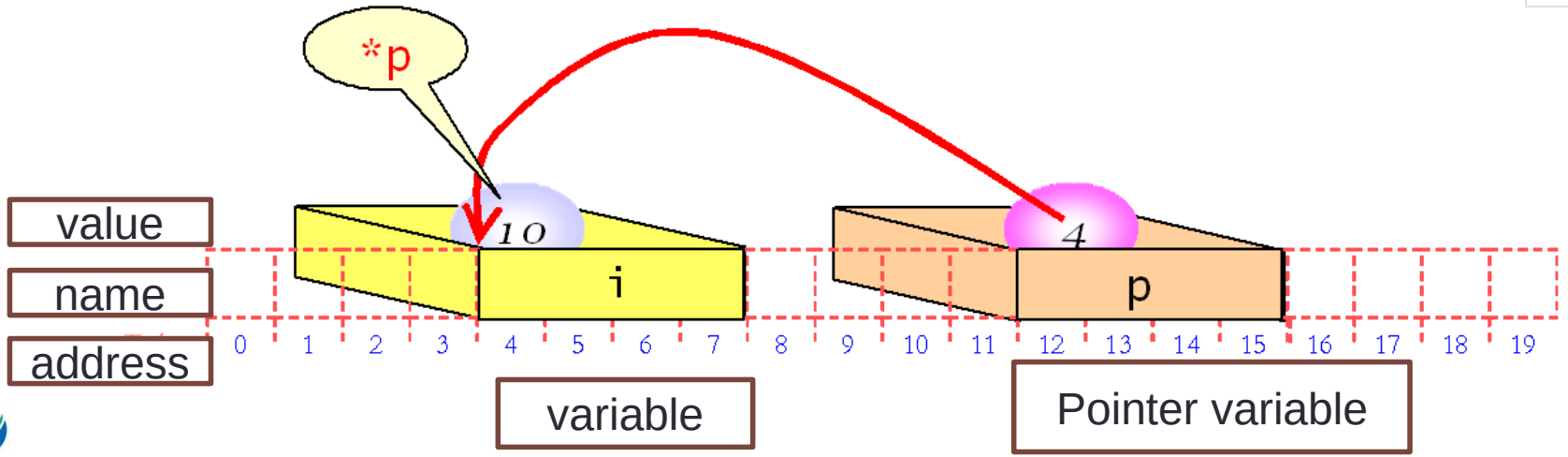
```
int y = 5;
int *yPtr;
yPtr = &y;
```



Pointer operators

- Indirection/dereferencing operator :*
  - ✓Return the value at the pointed address

```
int i = 10;
int *p = &i;
printf("%d\n", *p);     // 10
*p = 20;
printf("%d\n", *p);     // 20
```



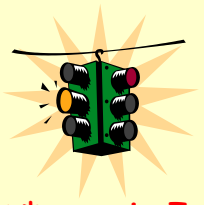
EX #1 pointer and address

```
1  ////////////////////////////////////////////////////////////////////////// ex1
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int n = 10;
7      int *m;
8      m = &n;
9      printf("address: %d, value: %d\n",m,*m);
10
11     n=100;
12     printf("address: %d, value: %d\n",m,*m);
13     return 0;
14 }
```

```
address: 2096448, value: 10
address: 2096448, value: 100
계속하려면 아무 키나 누르십시오 . . .
```

EX #2 pointer

```
#include <stdio.h>

int main(void)
{
    char c = 'A';
    int i = 10000;
    double d = 6.78;

    char *pc = &c;          // char type pointer
    int *pi = &i;           // int type pointer
    double *pd = &d;        // double type pointer

    (*pc)++;                // increased by 1 using indirection operator
    *pi = *pi + 1;          // increased by 1 using indirection operator
    *pd += 1;               // increased by 1 using indirection operator

    printf("c = %c\n", c);
    printf("i = %d\n", i);
    printf("d = %f\n", d);

    return 0;
}
```

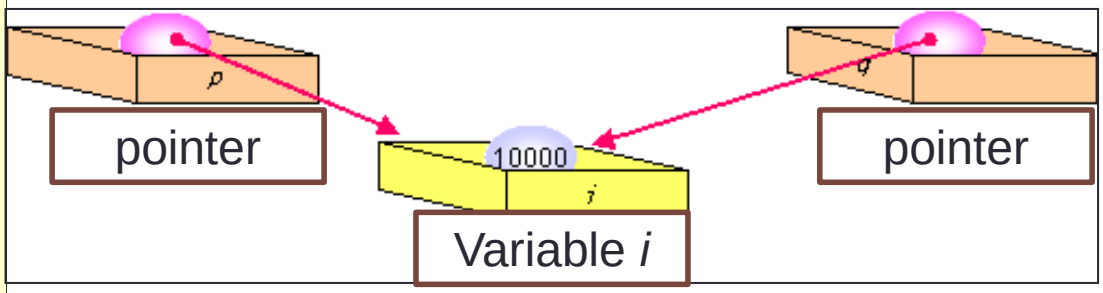


*pc++ is Incorrect

```
c = B
i = 10001
d = 7.780000
```

EX #3 pointer

```
#include <stdio.h>

int main(void)
{
	int i = 10000;
	int *p, *q;

	p = &i;
	q = &i;

	*p = *p + 1;
	*q = *q + 1;

	printf("i = %d\n", i);

	return 0;
}
```




```
i = 10002
```

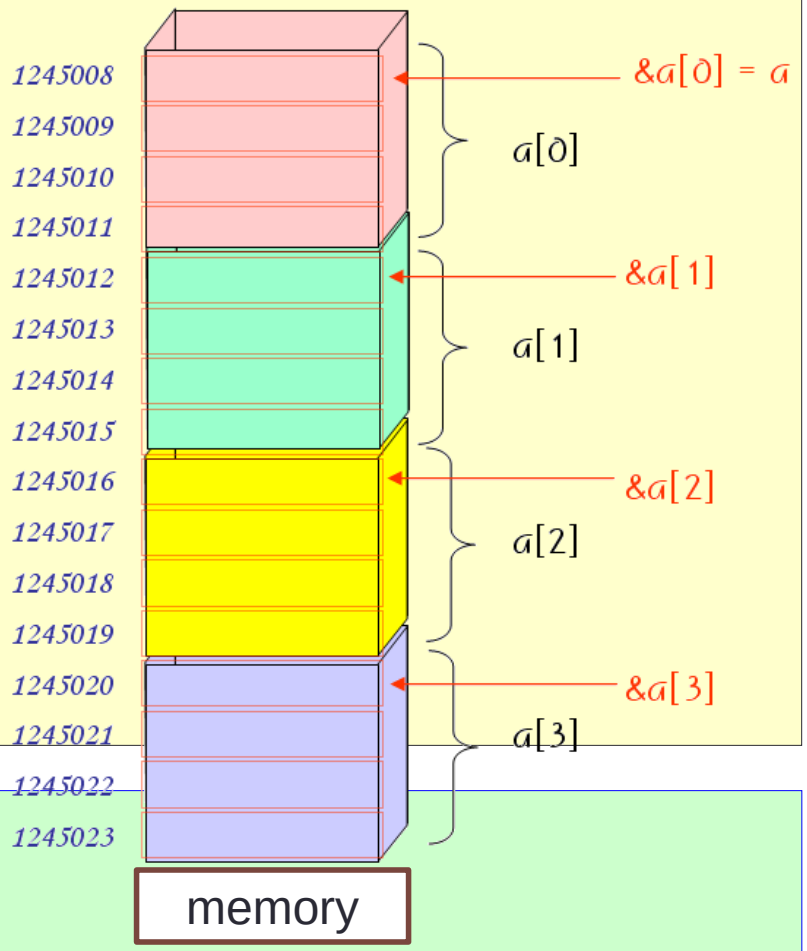
EX #4 pointer

- The pointer and variable data type must be same

```
#include <stdio.h>

int main(void)
{
	int i;
	double *pd;

	pd = &i;		// error! Pointer type is double but variable type is int
	*pd = 36.5;

	return 0;
}
```

EX #5 pointer and array

```
// relationship between the pointer and the array
#include <stdio.h>

int main(void)
{
    int a[] = { 10, 20, 30, 40, 50 };

    printf("&a[0] = %u\n", &a[0]);
    printf("&a[1] = %u\n", &a[1]);
    printf("&a[2] = %u\n", &a[2]);

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

    return 0;
}
```

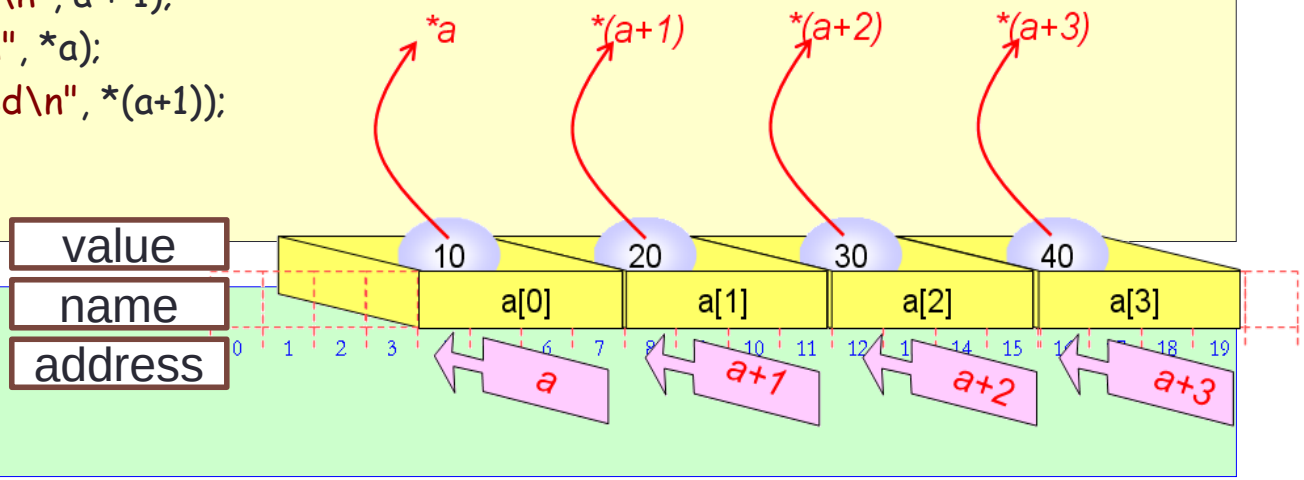



```
&a[0] = 1245008
&a[1] = 1245012
&a[2] = 1245016
a = 1245008
```

EX #6 pointer and array

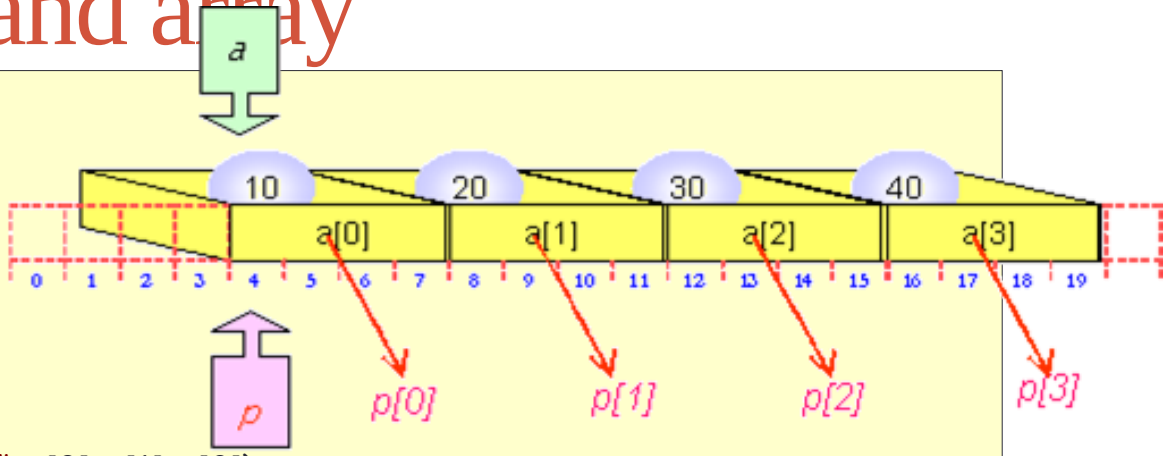
```
// relationship between the pointer and the array
#include <stdio.h>

int main(void)
{
      int a[] = { 10, 20, 30, 40, 50 };

      printf("a = %u\n", a);
      printf("a + 1 = %u\n", a + 1);
      printf("*a = %d\n", *a);
      printf("*(a+1) = %d\n", *(a+1));
      return 0;
}
```

```
a = 1245008
a + 1 = 1245012
*a = 10
*(a+1) = 20
```



EX #7 pointer and array

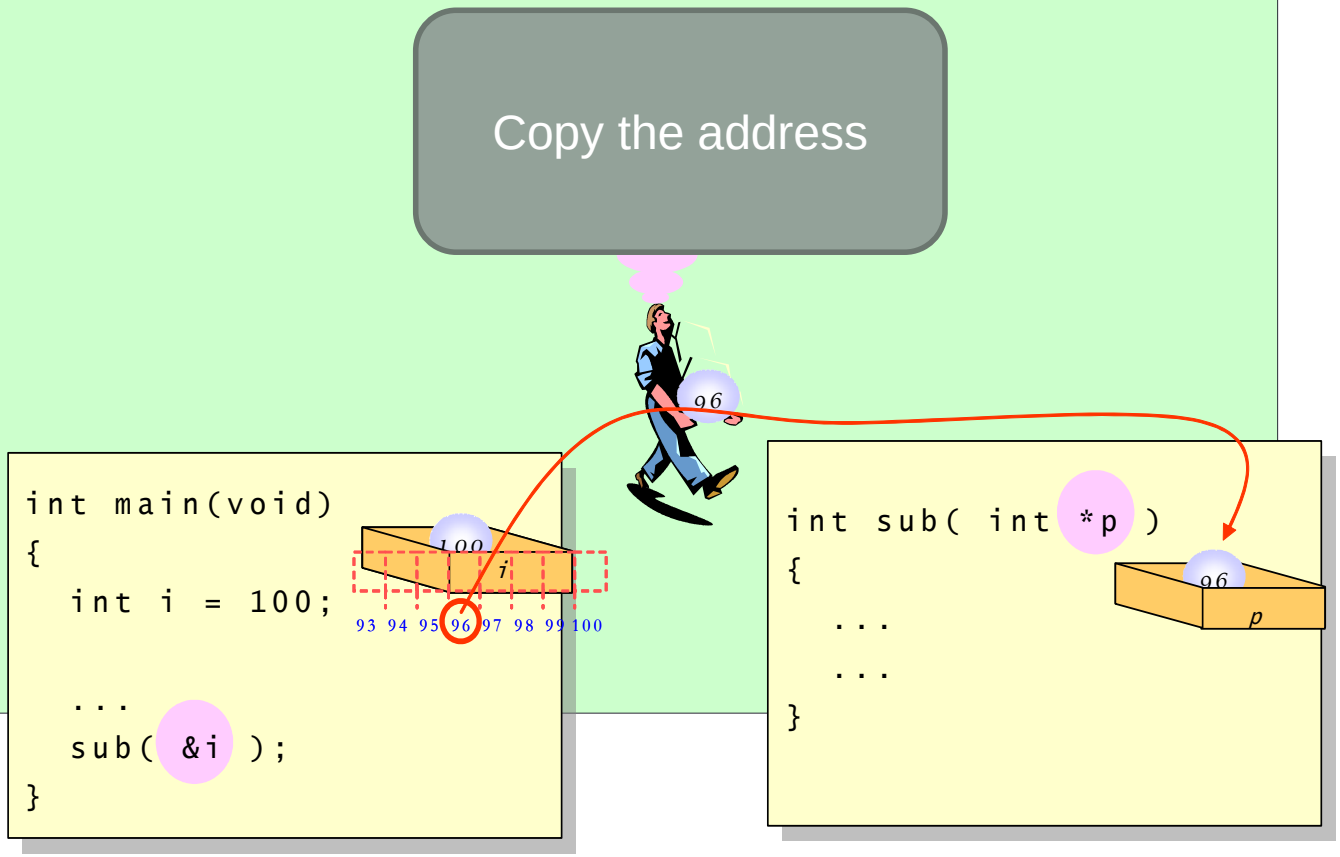
```
#include <stdio.h>

int main(void)
{
	int a[] = { 10, 20, 30, 40, 50 };
	int *p;

	p = a;
	printf("a[0]=%d a[1]=%d a[2]=%d \n", a[0], a[1], a[2]);
	printf("p[0]=%d p[1]=%d p[2]=%d \n\n", p[0], p[1], p[2]);

	p[0] = 60;
	p[1] = 70;
	p[2] = 80;

	printf("a[0]=%d a[1]=%d a[2]=%d \n", a[0], a[1], a[2]);
	printf("p[0]=%d p[1]=%d p[2]=%d \n", p[0], p[1], p[2]);

	return 0;
}
```




```
a[0]=10 a[1]=20 a[2]=30
p[0]=10 p[1]=20 p[2]=30

a[0]=60 a[1]=70 a[2]=80
p[0]=60 p[1]=70 p[2]=80
```

EX #8 call by reference

```
#include <stdio.h>
void sub(int *p);

int main(void)
{
      int i = 100;

      sub(&i);
      return 0;
}

void sub(int *p)
{
      *p = 200;
}
```

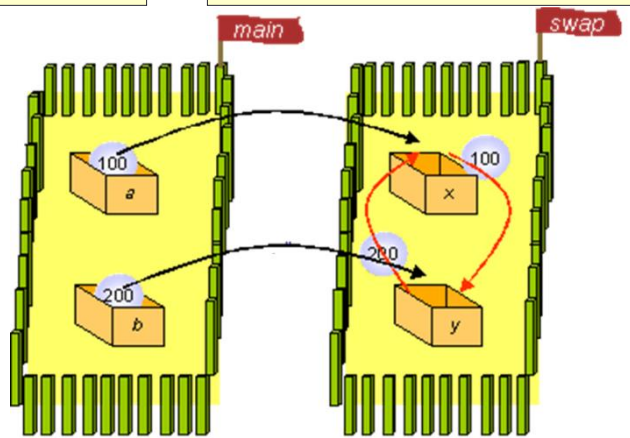


EX #9 call by value -swap

```c
#include <stdio.h>
void swap(int x, int y);
int main(void)
{
	int a = 100, b = 200;
	printf("main() a=%d b=%d\n",a, b);

	swap(a, b);

	printf("main() a=%d b=%d\n",a, b);
	return 0;
}
```

```c
void swap(int x, int y)
{
	int tmp;

	printf("swap() x=%d y=%d\n",x, y);

	tmp = x;
	x = y;
	y = tmp;

	printf("swap() x=%d y=%d\n",x, y);
}
```

```
main() a=100 b=200
swap() x=100 y=200
swap() x=200 y=100
main() a=100 b=200
```

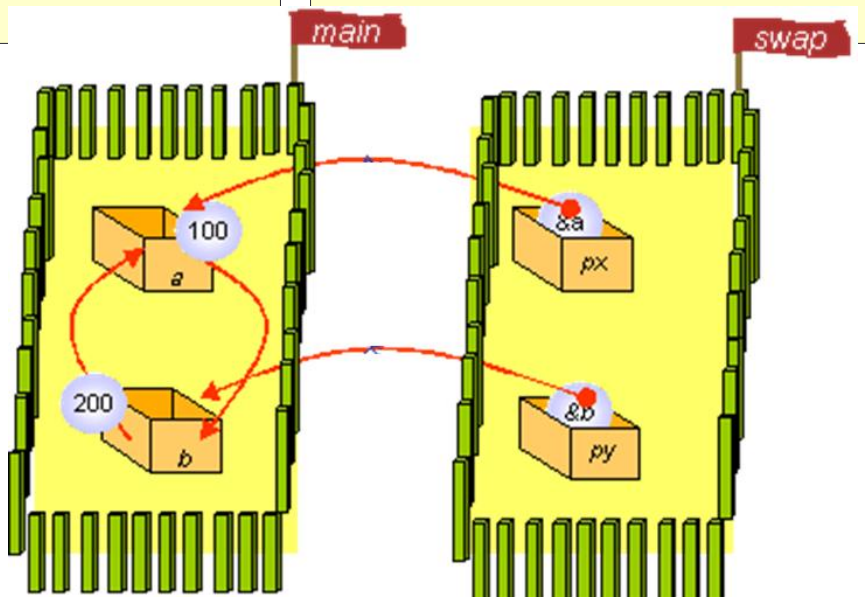


EX #10 call by reference-swap

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

	swap(&a, &b);

	printf("main() a=%d b=%d\n",a, b);
	return 0;
}
```

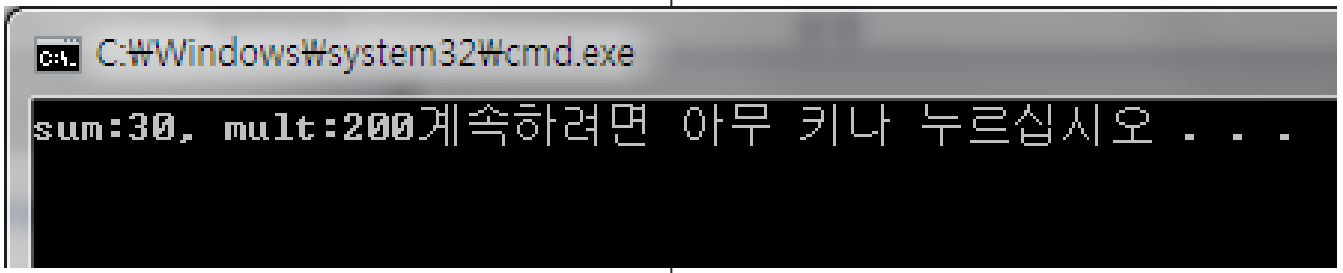
```c
void swap(int *px, int *py)
{
	int tmp;
	printf("swap() *px=%d *py=%d\n", *px, *py);

	tmp = *px;
	*px = *py;
	*py = tmp;

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

```
main() a=100 b=200
swap() *px=100 *py=200
swap() *px=200 *py=100
main() a=200 b=100
```



EX #11 More than 2 return values

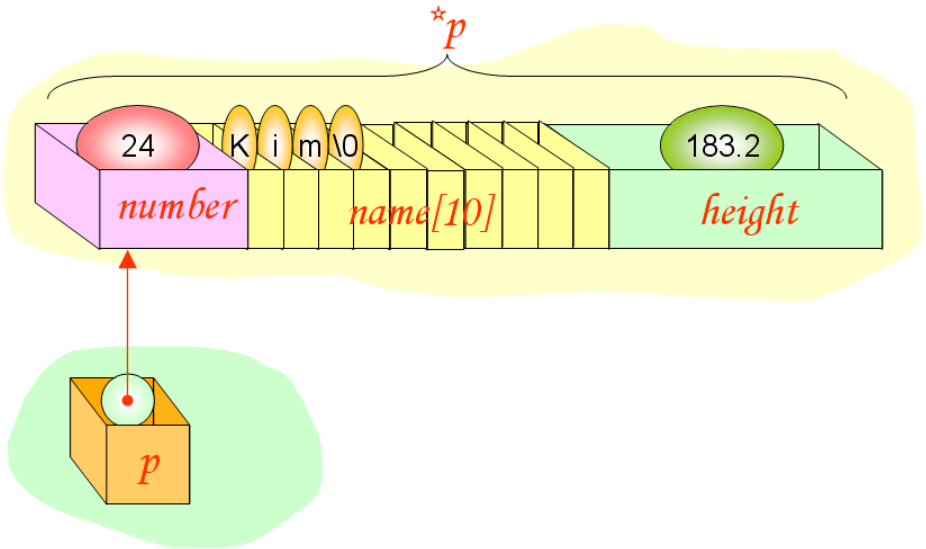
```
1  ////////////////////////////////////////////////////////////////////// ex10
2  #include <stdio.h>
3
4  void addmult(int a, int b, int *sum, int *mult )
5  {
6      *sum = a + b;
7      *mult = a*b;
8  }
9  int main(void)
10 {
11     int x = 10;
12     int y = 20;
13     int s;
14     int m;
15
16     addmult(x,y,&s,&m);
17     printf("sum:%d, mult:%d",s,m);
18 }
```



Structure and pointer

- The pointer point to structure

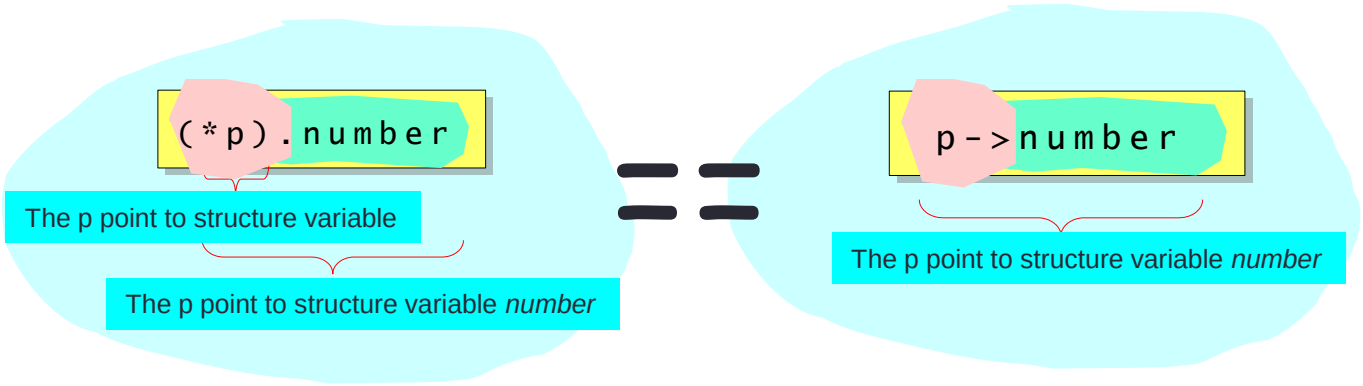
```
struct student s = { 20070001, "hong", 180.2 };
struct student *p;

p = &s;

printf("SID=%d Name=%s Height=%f \n", s.number, s.name, s.height);
printf("SID=%d Name=%s Height=%f \n", (*p).number,(*p).name,(*p).height);
```



-> Operator

- "->" operator can be used when the structure pointer refer to the structure member

```
struct student s = { 20070001, "hong", 180.2 };
struct student *p;

p = &s;

printf("SID=%d Name=%s Height=%f \n", s.number, s.name, s.height);
printf("SID=%d Name=%s Height=%f \n", (*p).number,(*p).name,(*p).height);
printf("SID=%d Name=%s Height=%f \n", p->number, p->name, p->height);
```



EX #12: refer the structure using pointer

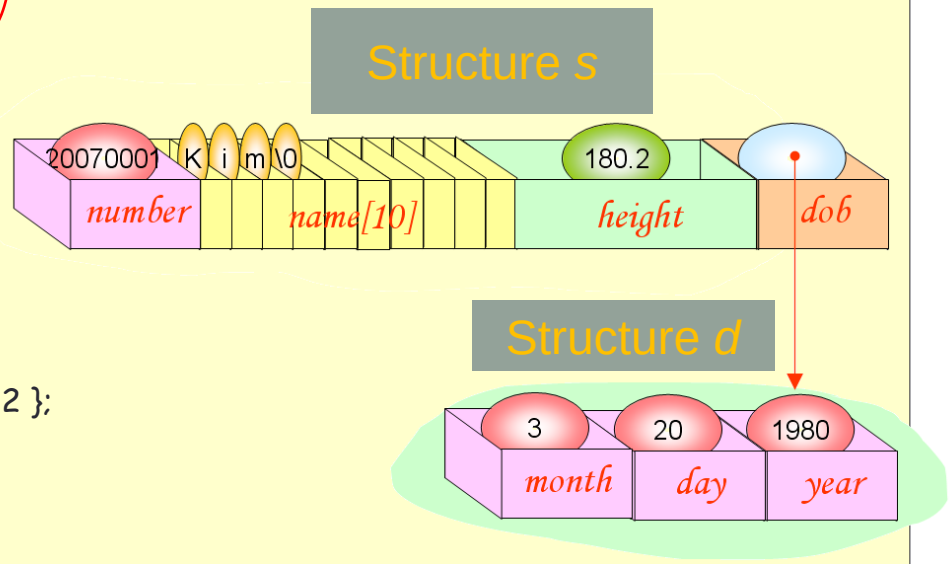
```c
// 포인터를 통한 구조체 참조
#include <stdio.h>

struct student {
	int number;
	char name[20];
	double height;
};

int main(void)
{
	struct student s = { 20070001, "홍길동", 180.2 };
	struct student *p;

	p = &s;

	printf("학번=%d 이름=%s 키=%f \n", s.number, s.name, s.height);
	printf("학번=%d 이름=%s 키=%f \n", (*p).number,(*p).name,(*p).height);
	printf("학번=%d 이름=%s 키=%f \n", p->number, p->name, p->height);

	return 0;
}
```

```
학번=20070001 이름=홍길동 키=180.200000
학번=20070001 이름=홍길동 키=180.200000
학번=20070001 이름=홍길동 키=180.200000
```

EX #13:Point to a structure member

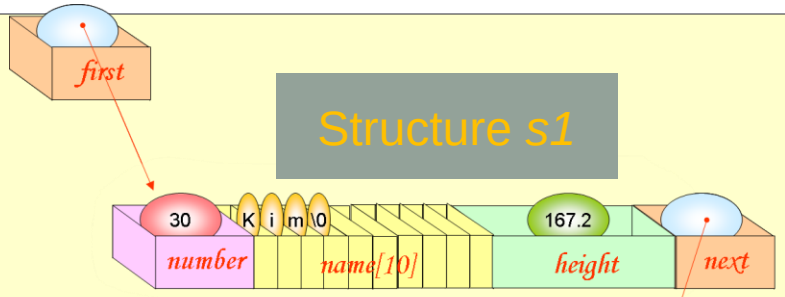
```
struct date {
        int month;
        int day;
        int year;
};
struct student {
        int number;
        char name[20];
        double height;
        struct date *dob;
};
int main(void)
{
        struct date d = { 3, 20, 1980 };
        struct student s = { 20070001, "Kim", 180.2 };

        s.dob = &d;

        printf("SID: %d\n", s.number);
        printf("Name: %s\n", s.name);
        printf("Height: %f\n", s.height);
        printf("Bday: %d. %d. %d.\n", s.dob->year, s.dob->month, s.dob->day);
        return 0;
}
```

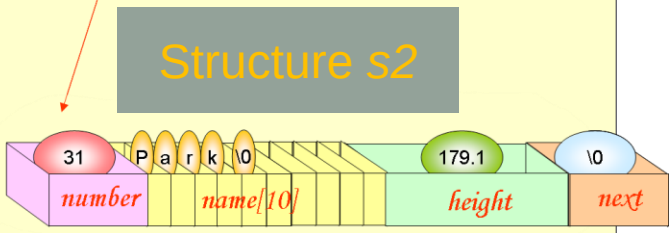
SID: 20070001
Name: Kim
Height: 180.200000
Bday: 1980. 3. 20.



EX #14 : Self-referential structure

```
struct student {
        int number;
        char name[10];
        double height;
        struct student *next;
};

int main(void)
{
        struct student s1 = { 30, "Kim", 167.2, NULL };
        struct student s2 = { 31, "Park", 179.1, NULL };
        struct student *first = NULL;
        struct student *current = NULL;

        first = &s1;
        s1.next = &s2;
        s2.next = NULL;

        current = first;
        while( current != NULL )
        {
                printf("SID=%d Name=%s, Height=%f\n", current->number,
                                current->name, current->height);
                current = current->next;
        }
}
```






```
SID =30 Name=Kim, Height=167.200000
SID=31  Name=Park, Height=179.100000
```

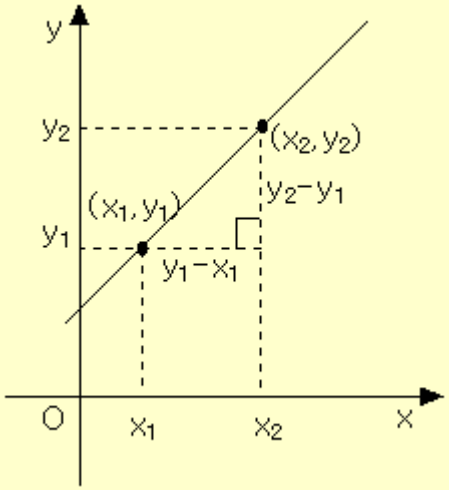
Structure and function

- If the input data type is the structure pointer
  - ✓ Save the time and memory space

```
int equal(struct student const *p1, struct student const *p2)
{
      if( strcmp(p1->name, p2->name) == 0 )
            return 1;
      else
            return 0;
}
```

EX #15 : slope and Intercept

```
#include <stdio.h>
struct point {
        int x;
        int y;
};
// compute the slope and intercept
int get_line_parameter(struct point p1, struct point p2, float *slope, float *yintercept)
{
        if( p1.x == p2.x )
                return (-1);
        else
        {
          *slope = (float)(p2.y - p1.y)/(float)(p2.x - p1.x);
          *yintercept = p1.y - (*slope) * p1.x;
          return (0);
        }
}
int main(void)
{
        struct point pt1 = {3, 3}, pt2 = {6, 6};
        float s,y;

        if( get_line_parameter(pt1, pt2, &s, &y) == -1 )
                printf("error: same x axis.\n");
        else
                printf("Gradient is %f, y-intercept is %f\n", s, y);
        return 0;
}
```




```
Gradient is 1.000000, y-intercept is 0.000000
```

HW #9

- Execute the EX#1~15, Then copy the source **code(with comment**) and results to the report.