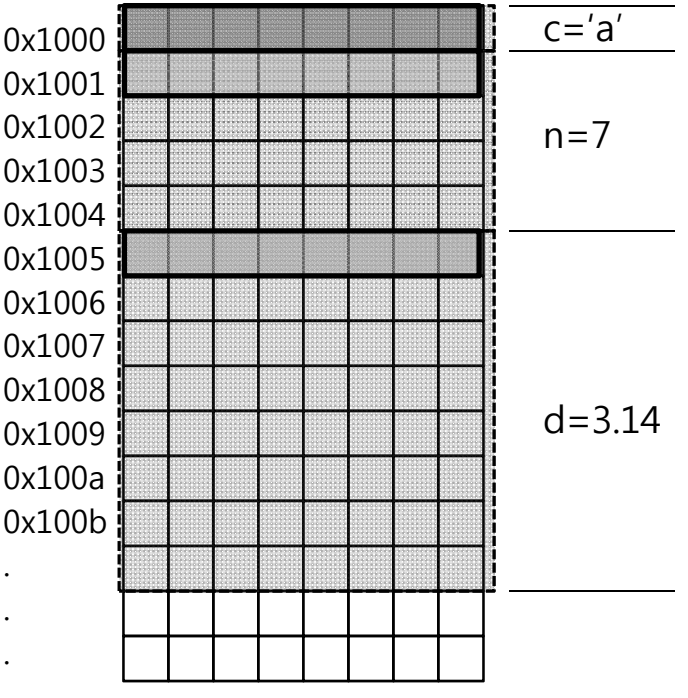
18. 포인터

18-1) 포인터의 이해

포인터와 포인터 변수

- 포인터는 메모리에 있는 데이터의 주소를 가지고 있는 변수
- "포인터"를 흔히 "포인터 변수"라 한다.
- 포인터는 변수이지만 데이터의 값을 가지고 있지 않고, 데이터의 주소를 가지고 있다.

```
int main(void)
{
        char c = 'a';
        int n = 7;
        double d = 3.14;
        ........
}
```

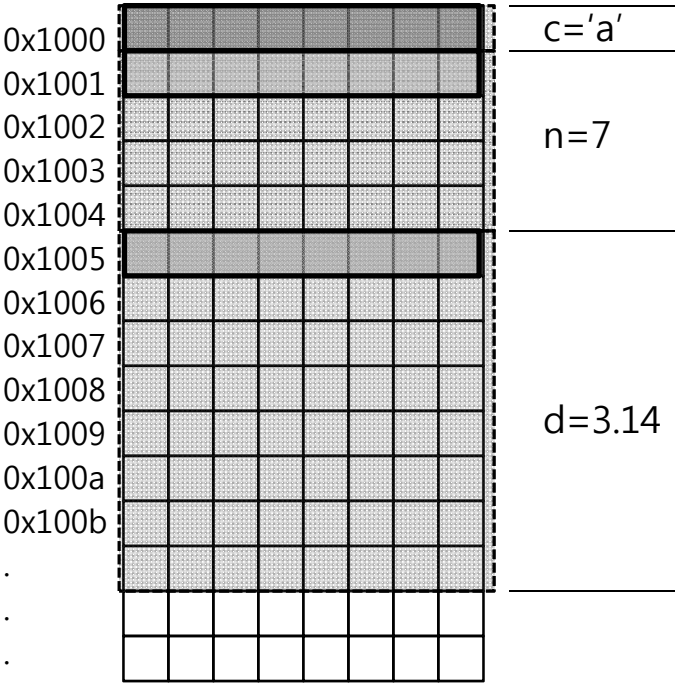


18-1) 포인터의 이해

- **포인터와 포인터 변수**

```
int main(void)
{
        char c = 'a';
        int n = 7;
        double d = 3.14;
         .......
}
```

```
&c = 0x1000
&n = 0x1001
&d = 0x1005
```



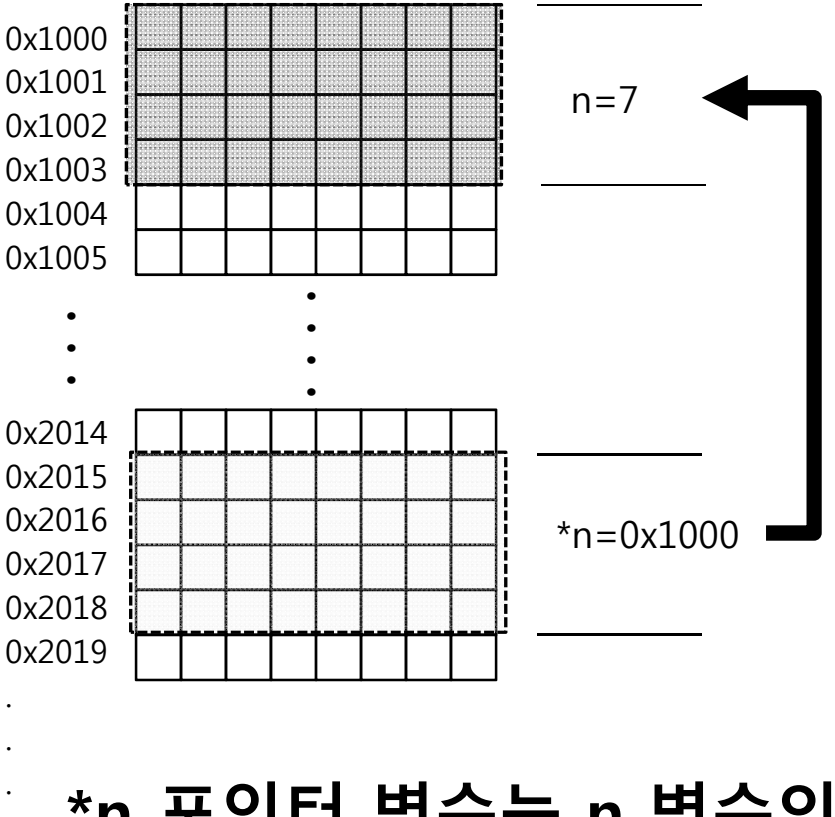
18-1) 포인터의 이해 (예제 18-1)

```
#include <stdio.h>

int main(void)
{
        char c = 'a';
        int n = 7 ;
        double d = 3.14 ;

        printf("c의 주소 :%p₩n",&c);
        printf("n의 주소 :%p₩n",&n);
        printf("d의 주소 :%p₩n",&d);
 return 0;
}
```

18-1) 포인터의 이해

포인터의 이해

- 32 Bits 시스템 기반 : 4 Bytes



- 8 Bits 시스템 기반 : 1 Bytes
- 16 Bits 시스템 기반 : 2 Bytes
- 32 Bits 시스템 기반 : 4 Bytes
- 64 Bits 시스템 기반 : 8 Bytes

***n 포인터 변수는 n 변수의 주소를 가리킨다.**

18-1) 포인터의 이해

포인터의 타입과 선언

- 포인터 선언 시 사용되는 연산자 : *(에스크립터)

-포인터를 선언하려면 포인터가 가리키게 되는 대상의 형을 먼저 쓰고 *를 붙인 다음, 포인터의 이름을 쓴다.

```
int main(void)
{
        int *p;          // p 라는 이름의 int 형 포인터
        char *pb;        // pb라는 이름의 char 형 포인터
        double *pc:      // pc라는 이름의 double형 포인터
        ........
}
```

int* p; ⇨ int * p; ⇨ int *p;

1. 포인터의 이해

포인터의 타입과 선언

- 포인터 선언 시 사용되는 연산자 : *(에스크립터)
- A 형 포인터(A*) : A형 변수의 주소 값을 저장.

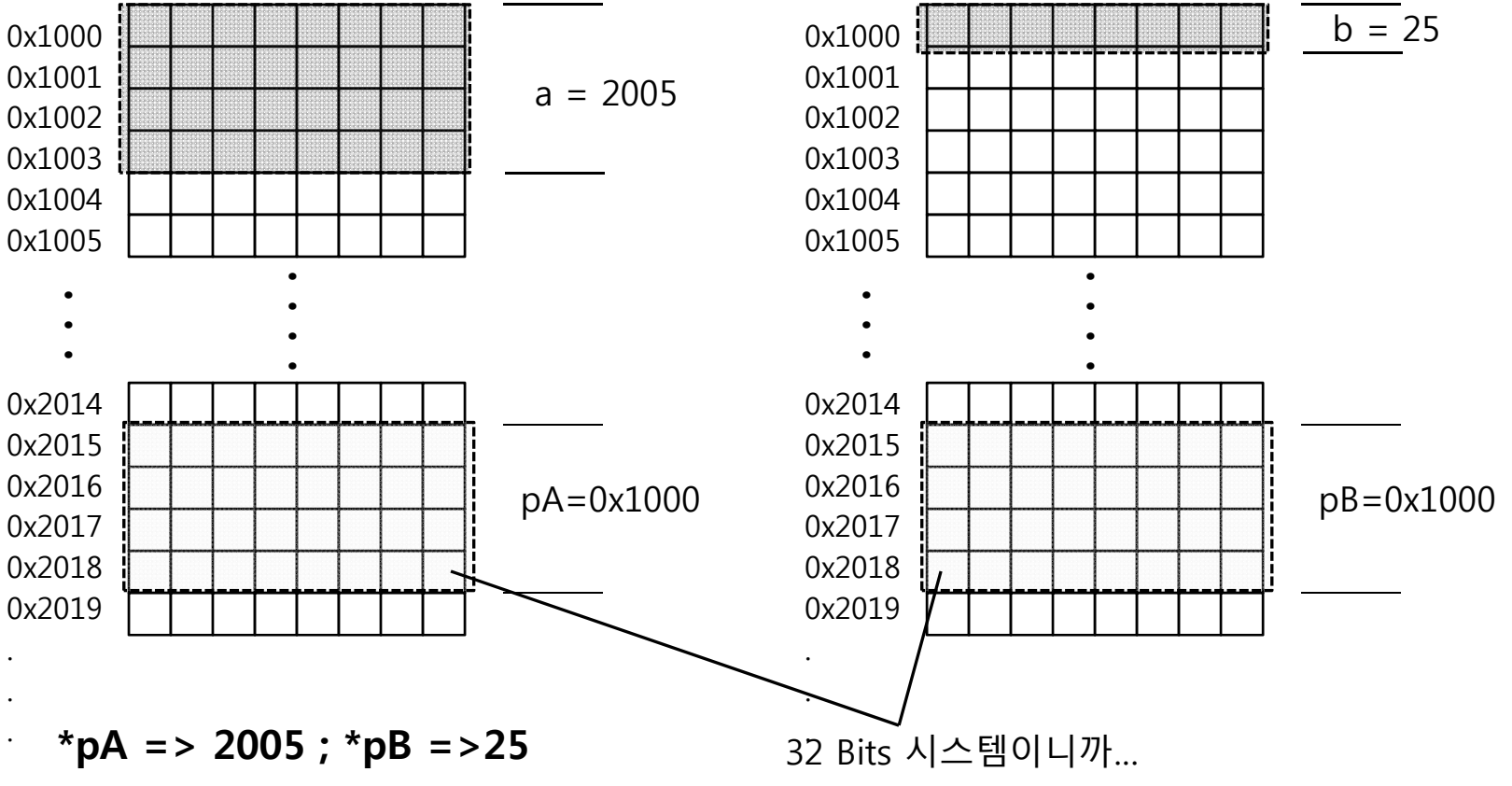
```
int main(void)
{
        int *a;          // a 라는 이름의 int 형 포인터
        char *b;         // b라는 이름의 char 형 포인터
        double *c:       // c라는 이름의 double형 포인터
        ........
}
```

int* a; = int * a; = int *a;

1. 포인터의 이해

```
int a = 2005 ;
int *pA ;
pA = &a ;
```

```
char b = 25 ;
char *pB ;
pB = &b ;
```



1. 포인터의 이해

포인터 관련 연산자

- &(엔퍼센트) 연산자 : 변수의 주소값 반환
- *(에스크립터) 연산자 : 포인터가 가리키는 메모리 참조.

```
int main(void)
{
  int a = 2005;
  int *pA ;

  pA= &a;
  printf("%d", a);  // 직접 접근
  printf("%d", *pA);  //간접접근
  ……..
}
```

18-1) 포인터의 이해 (예제 18-2)

```
#include <stdio.h>

int main(void)
{
        int a =2005;

        // int *pA = &a ;
        int *pA;
        pA = &a;

        printf("%d  ",a);
        printf("%d  ",*pA);
 return 0 ;
}
```

1. 포인터의 이해

- * 연산자

a * b 곱셈

int* p 포인터 연산자 선언

18-1) 포인터의 이해 (예제 18-3)

```
#include <stdio.h>

int main(void)
{
        int a ;
        int *p = NULL ;
        p = &a ;

        scanf("%d", &a);
        printf("a = %d, *p = %d₩n",a,*p);

        scanf("%d",p);
        printf("a = %d, *p = %d₩n",a,*p);

        a = a+5 ;
        printf("a = %d, *p = %d₩n",a,*p);
```

18-1) 포인터의 이해 (예제 18-3)

```
        *p = *p * 2 ;
        printf("a = %d, *p = %d₩n",a,*p);

        (*p)--;
        printf("a = %d, *p = %d₩n",a,*p);

        return 0 ;
}
```

18-1) 포인터의 이해 (예제 18-4)

- **포인터에 다양한 타입이 존재하는 이유**

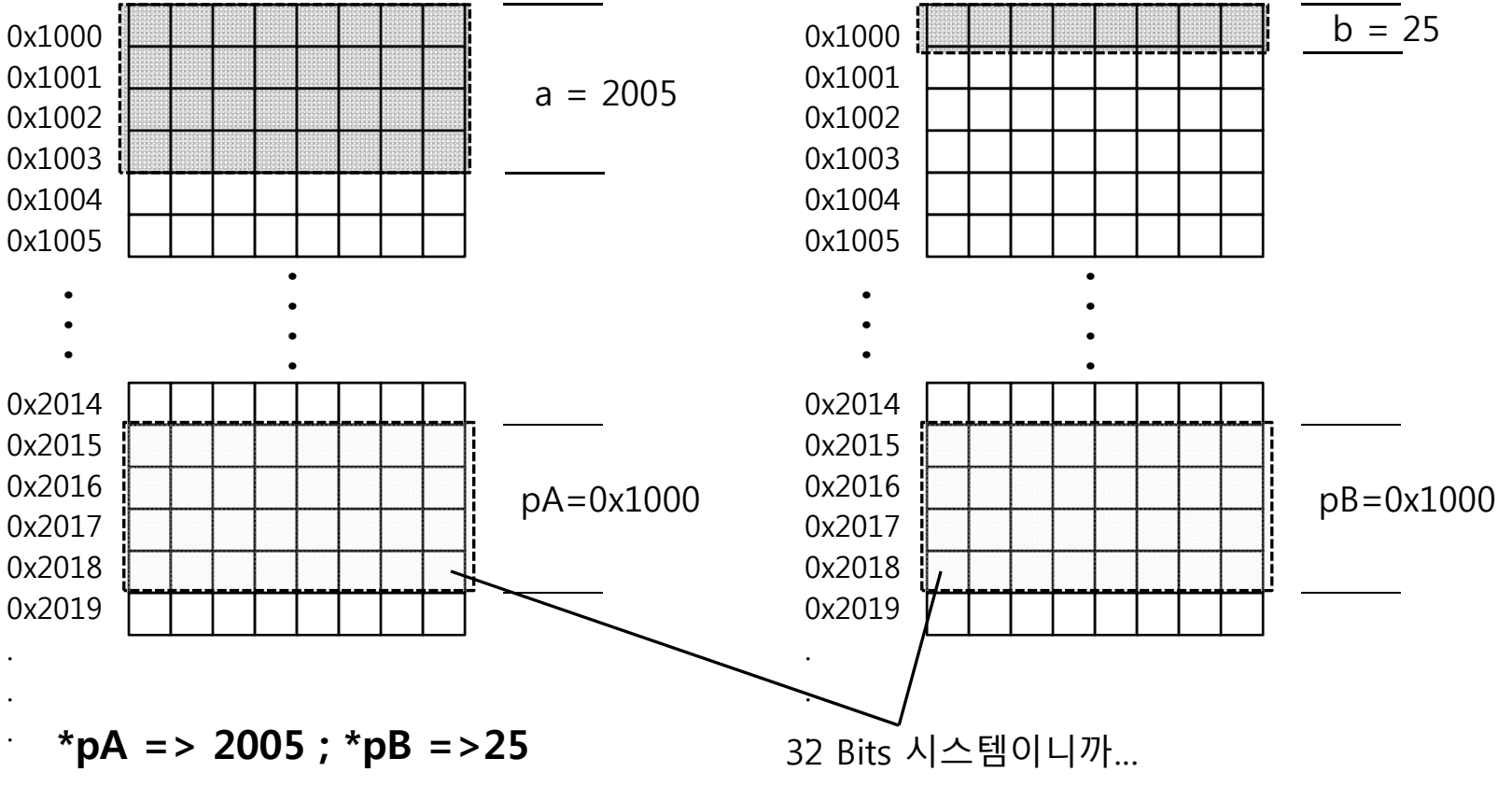
  - 포인터 타입은 참조할 메모리의 크기 정보를 제공

```
#include <stdio.h>

int main(void)
{
        int a = 10 ;
        int *pA = &a ; // int  *pA =NULL;  pA = &a;

        double e = 3.14 ;
        double *pE = &e ; //double *pE =NULL; pE= &e

        printf("%d  %f ₩n",*pA, *pE);
        return 0 ;
}
```

18-1) 포인터의 이해

```
int a = 2005 ;
int *pA ;
pA = &a ;
```

```
char b = 25 ;
char *pB ;
pB = &b ;
```



***pA => 2005 ; *pB =>25**

32 Bits 시스템이니까...

18-1) 포인터의 이해 (예제 18-5)

```
#include <stdio.h>

int main(void)
{
        int i = 1000;
        int *p = NULL;

        p = &i;

        printf(" i = %d₩n",i);
        printf("&i = %p₩n₩n",&i);

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

        return 0 ;
}
```

18-1) 포인터의 이해 (예제 18-6)

```
#include <stdio.h>

int main(void)
{
        int a=10, b=20;
        int *p =NULL;

        p = &a ;
        printf("p = %p₩n",p);
        printf("*p = %d₩n₩n",*p);

        p = &b ;
        printf("p = %p₩n",p);
        printf("*p = %d₩n₩n",*p);

return 0 ;
}
```

18-1) 포인터의 이해 (예제 18-7)

```
#include <stdio.h>

int main(void)
{
        int i = 10;
        int *p=NULL;

        p = &i;
        printf("i = %d₩n",i);

        *p = 20 ;
        printf("i = %d₩n",i);

        return 0 ;
}
```

18-2) 잘못된 포인터의 사용

- **사례 1**

```
int main(void)
{
  int *pA ;  // 쓰레기 값으로 초기화
  *pA = 10; // 아주 위험한 코드
   ........
}
```

- **사례 2**

```
int main(void)
{
  int *pA = 10000 ;  // 10000 이 어디인데?
  *pA = 123;  // 10000번지에 123이 저장
   ........
}
```

18-2) 잘못된 포인터의 사용

- 포인터가 아무것도 가리키지 않을 때는 NULL 설정

```
int main(void)
{
  int *p = NULL ;
.....
}
```

18-2) 잘못된 포인터의 사용

- 포인터타입과 변수의 타입은 일치

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

 pd = & i;
*pd = 36.5;
return 0;
}
```

18-3) 포인터 연산

- 포인터의 증감 연산 시 증가되는 값

포인터 타입	++연산 후 증가되는 값
char	1
short	2
int	4
float	4
double	8

18-4) 포인터 연산 (예제 18-8)

```
#include <stdio.h>

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

  pc = (char *) 10000;
  pi = (int *) 10000;
  pd = (double *) 10000;

  printf("증가전 pc = %d, pi = %d,   pd = %d₩n",pc,pi,pd);
```

18-4) 포인터 연산 (예제 18-8)

```
 pc++;
  pi++;
  pd++ ;
  printf("증가후 pc = %d, pi = %d,    pd = %d₩n",pc,pi,pd);

  printf("pc+2 = %d, pi+2 = %d, pd+2 = %d₩n",pc+2,pi+2,pd+2);

return 0;
}
```

18-4) 포인터 연산

*p++; // p가 가리키는 위치에서 값을 가져온 후 p를 증가

(*p)++ ; // p가 가리키는 위치의 대상의 값을 증가

수식	의미
v = *p++	p가 가리키는 값을 v에 대입한 후에 p를 증가
v = (*p)++	p가 가리키는 값을 v에 대입한 후에 p가 가리키는 값을 증가
v =*++p	p를 증가시킨 후에 p가 가리키는 값을 v에 대입
v = ++*p	p가 가리키는 값을 가져온 후에 그 값을 증가하여 v에 대입

18-4) 포인터 연산 (예제 18-9)

```
#include <stdio.h>

int main(void)
{
    int i = 10;
    int *pi = &i ;

    printf("i =%d, pi = %p₩n",i,pi);
    (*pi)++;

    printf("i =%d, pi = %p₩n",i,pi);
    *pi++;

    printf("i =%d, pi = %p₩n",i,pi);

    return 0;
}
```

18-5) 포인터의 형 변환 (예제 18-10)

```
#include <stdio.h>

int main(void)
{
        char buff[4];
        int  *pi;
        char *pc ;

        pi = (int *) buff;
        *pi = 0x1234;

        printf("%x₩n",*pi);

        pc = (char *)buff;
        printf("%x  %x ₩n", *pc, *(pc+1));
        return 0;
}
```

18-6) 포인터와 배열 (예제 18-11)

```
#include <stdio.h>

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

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

        printf("a = %p₩n",a);

        return 0;
}
```

18-6) 포인터와 배열 (예제 18-12)

```
#include <stdio.h>

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

        printf("a = %x₩n",a);
        printf("a + 1 = %x₩n",a+1);
        printf("*a = %d₩n",*a);
        printf("*(a+1) = %d₩n",*(a+1));

        return 0;
}
```

18-6) 포인터와 배열 (예제 18-13)

```
#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",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;
}
```

18-7) 포인터와 함수 (예제 18-14 : 값에 의한 호출)

```
// 값에 의한 호출(call –by-value ): 복사본이 전달 된다.

#include <stdio.h>
void swap(int, int);

int main(void)
{
        int a =10, b = 20 ;

        printf("a = %d b = %d ₩n",a,b);
        swap(a,b);
        printf("a = %d b = %d ₩n",a,b);

        return 0;
}
```

18-7) 포인터와 함수 (예제 18-14 : 값에 의한 호출)

```
void swap(int x, int y)
{
        int tmp;
        printf("x = %d y = %d₩n",x,y);

        tmp = x;
        x = y;
        y = tmp;
        printf("x = %d y = %d₩n",x,y);
}
```

18-7) 포인터와 함수 (예제 18-15 : 참조에 의한 호출)

```
//  참조에 의한 호출(call –by-reference ): 원본이 전달 된다.

#include <stdio.h>
void swap(int *, int *);

int main(void)
{
        int a = 10, b = 20 ;

        printf("a= %d b = %d₩n",a,b);
        swap(&a, &b);
        printf("a= %d b = %d₩n",a,b);

 return 0 ;
}
```

18-7) 포인터와 함수 (예제 18-15 : 참조에 의한 호출)

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

        tmp = *px; // tmp = a;
        *px = *py ; //a = b
        *py = tmp ; // b= tmp
}
```

18-7) 포인터와 함수 (예제 18-16)

```
//포인터 인수를 통하여 결과를 반환하는 함수 .

#include <stdio.h>

void calculate(int, int, int *, int *);
int main(void)
{
 int x = 10, y =20 ;
 int sum = 0, diff = 0;

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

 printf("sum = %d diff = %d ₩n",sum,diff);
 return 0 ;
}
```

18-7) 포인터와 함수 (예제 18-16)

```
void calculate(int a, int b, int *s, int *df)
{
        *s = a +b ;
        *df = a - b ;
}
```

18. 연습문제

1. 2개의 정수의 합과 차, 곱셈과 나눗셈의 결과를 동시에 반환하는 함수를 작성하고 테스트하라. 단, 포인터 매개 변수를 사용한다.

```
void calculate(int x, int y, int *psum, int *pdiff, int *pmul, int *pdiv) {
                    . . .
}
```

2. 정수 배열의 원소들을 화면에 출력하는 함수를 작성하시오.
   출력 형식은 A[5] = { 1, 2, 3, 4, 5 } 와 같은 형식이 되도록 하라.

```
void print_array(int *A, int size);
```

18. 연습문제

3. 정수 배열 ArrayA[5]를 다른 정수 배열 ArrayB[5] 에 복사하는 함수를 작성하시오.

```
 void copy_array(int *A, int *B, int size)
{
        . . .
}
```