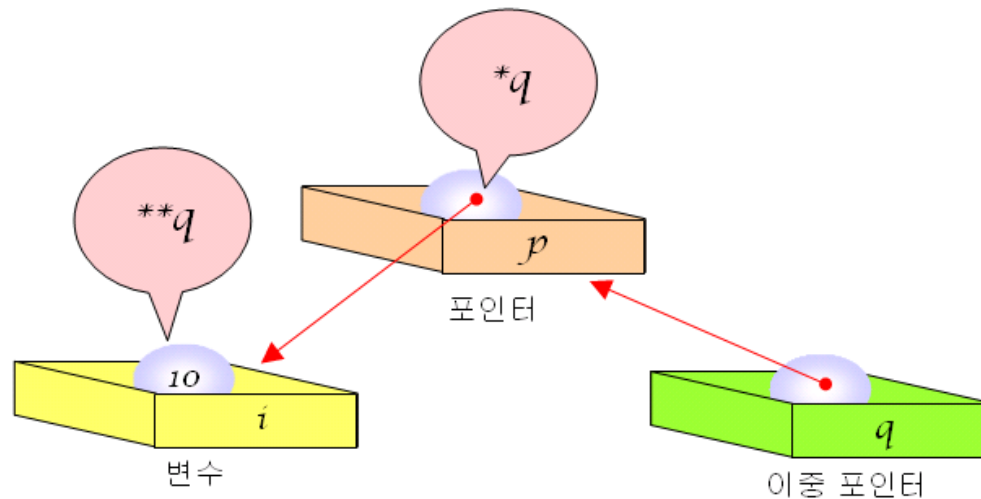


포인터 활용 Part 1

이중 포인터

- 이중 포인터(double pointer): 포인터를 가리키는 포인터
 - Pointer to Pointer

```
int i = 100;           // i는 int형 변수  
int *p = &i;           // p는 i를 가리키는 포인터  
int **q = &p;          // q는 포인터 p를 가리키는 이중 포인터
```



// 이중 포인터 프로그램

#include <stdio.h>

int main(void)

{

int i = 100;

int *p = &i;

int **q = &p;

*p = 200;

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

**q = 300;

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

return 0;

}

i=200	*p=200	**q=200
i=300	*p=300	**q=300

예제 #2

// 이중 포인터 프로그램

```
#include <stdio.h>
```

```
void set_proverb(char **q);
```

```
int main(void)
```

```
{
```

```
    char *s = NULL;
```

```
    set_proverb(&s);
```

```
    printf("selected proverb = %s\n",s);
```

```
    return 0;
```

```
}
```

```
void set_proverb(char **q)
```

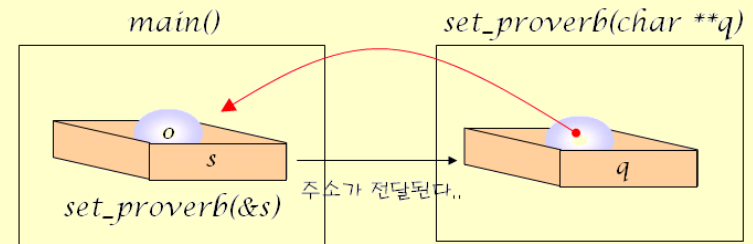
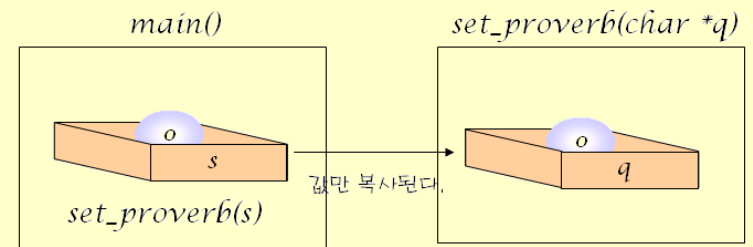
```
{
```

```
    static char *str1="A friend in need is a friend indeed";
```

```
    static char *str2="A little knowledge is a dangerous thing";
```

```
    *q = str1;
```

```
}
```

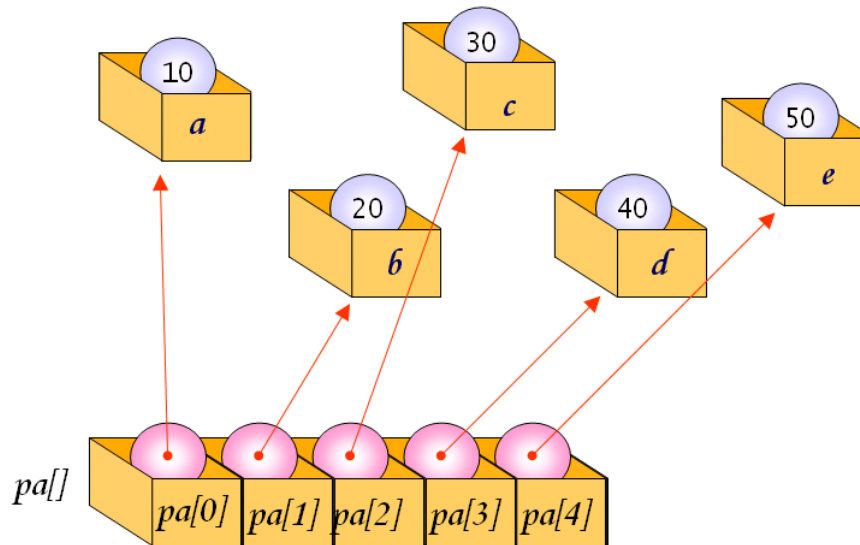


selected proverb = A friend in need is a friend indeed

포인터 배열 (Array of Pointers)

- 포인터 배열(array of pointers): 포인터들의 배열

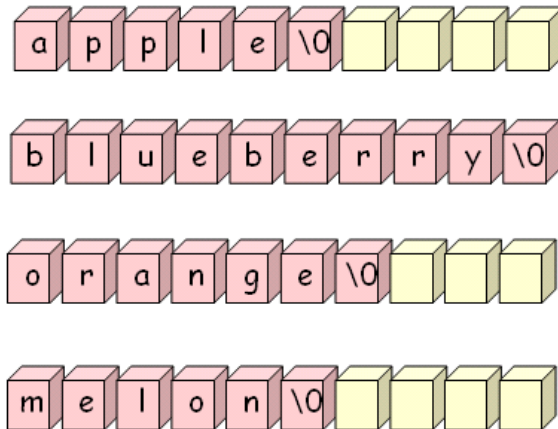
```
int a = 10, b = 20, c = 30, d = 40, e = 50;  
int *pa[5] = { &a, &b, &c, &d, &e };
```



문자열 배열 vs. 2차원 배열

- 문자열들을 저장하는 2차원 배열
 - 공간의 낭비가 발생할 수 있다.

```
char fname[ ][10] = {  
    "apple",  
    "blueberry",  
    "orange",  
    "melon"  
};
```



낭비되는
공간!

2차원 배열을 사
용하면 낭비되는
공간이 생성되죠.

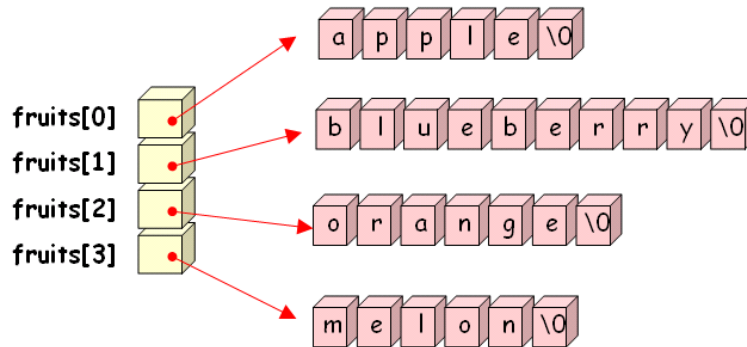


문자열 배열(Array of Strings)

■ 문자열 배열

- 가장 많이 사용되는 포인터 배열
- 문자열들을 효율적으로 저장할 수 있다.

```
char *fruits[ ] = {  
    "apple",  
    "blueberry",  
    "orange",  
    "melon"  
};
```



stringarray.c

```
// 문자열 배열
#include <stdio.h>

int main(void)
{
    int i, n;
    char *fruits[ ] = {
        "apple",
        "blueberry",
        "orange",
        "melon"
    };

    n = sizeof(fruits)/sizeof(fruits[0]);    // 배열 원소 개수 계산

    for(i = 0; i < n; i++)
        printf("%s\n", fruits[i]);

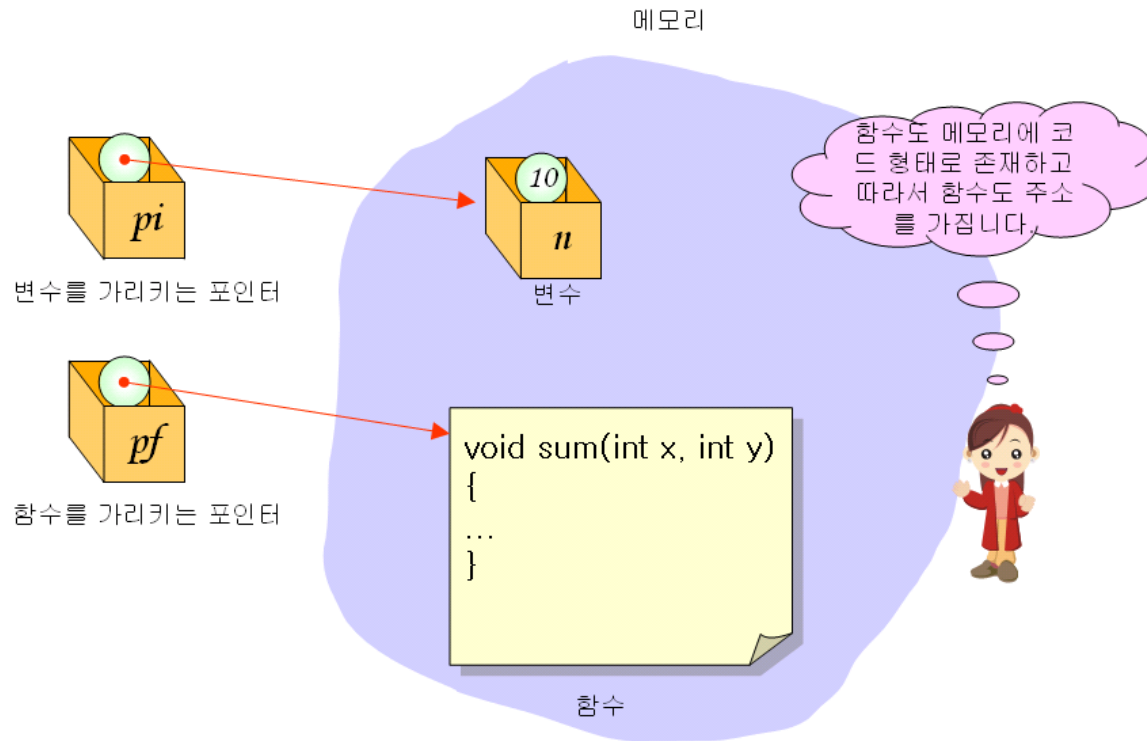
    return 0;
}
```

```
apple
blueberry
orange
melon
```


함수 포인터(Pointer to Function)

- 함수 포인터(function pointer): 함수를 가리키는 포인터

반환형 (*함수포인터이름)(매개변수1, 매개변수2, ...) ;



fp1.c

```
// 함수 포인터  
#include <stdio.h>
```

```
// 함수 원형 정의  
int add(int, int);  
int sub(int, int);
```

```
int main(void)  
{
```

```
    int result;  
    int (*pf)(int, int);
```

// 함수 포인터 정의

```
    pf = add;  
    result = pf(10, 20);  
    printf("10+20은 %d\n", result);
```

// 함수 포인터에 함수 add()의 주소 대입
// 함수 포인터를 통한 함수 add() 호출

```
    pf = sub;  
    result = pf(10, 20);  
    printf("10-20은 %d\n", result);
```

// 함수 포인터에 함수 sub()의 주소 대입
// 함수 포인터를 통한 함수 sub() 호출

```
    return 0;
```

```
}
```

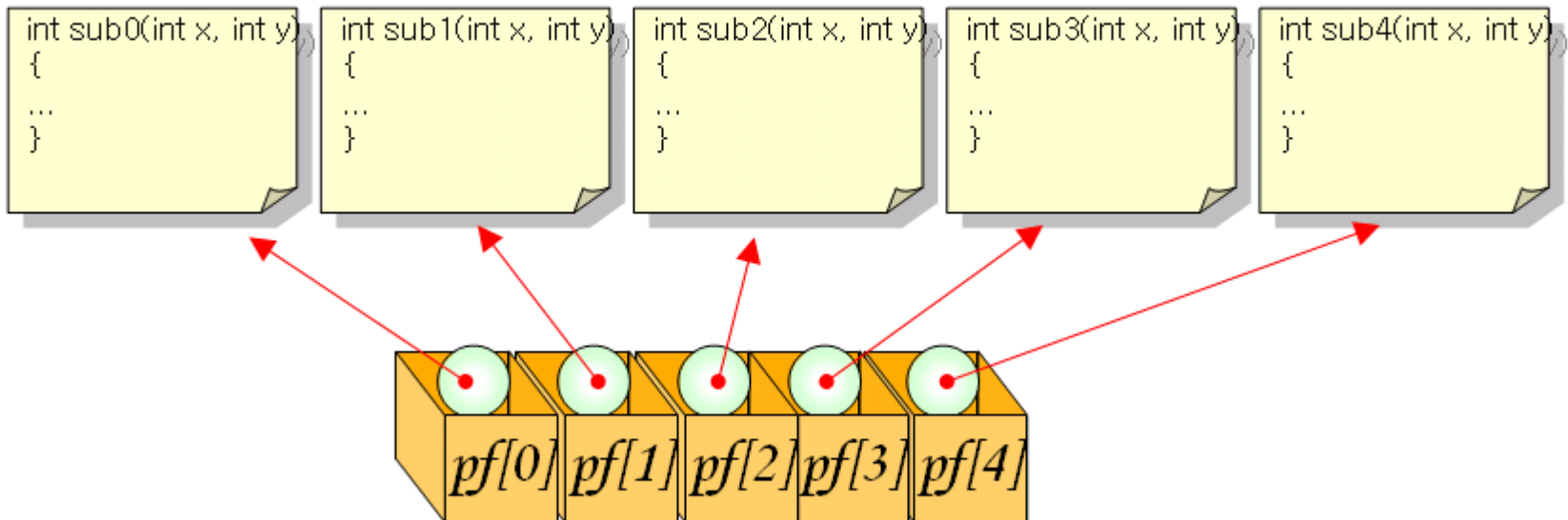
```
int add(int x, int y)  
{  
    return x+y;  
}
```

```
int sub(int x, int y)  
{  
    return x-y;  
}
```

10+20은 30
10-20은 -10

함수 포인터의 배열

- 반환형 (*배열이름[배열의_크기])(매개변수목록);
 - `int (*pf[5])(int, int);`



fp2.c

```
// 함수 포인터 배열
#include <stdio.h>

// 함수 원형 정의
void menu(void);
int add(int x, int y);
int sub(int x, int y);
int mul(int x, int y);
int div(int x, int y);

void menu(void)
{
    printf("=====\n");
    printf("0. 덧셈\n");
    printf("1. 뺄셈\n");
    printf("2. 곱셈\n");
    printf("3. 나눗셈\n");
    printf("4. 종료\n");
    printf("=====\n");
}
```

fp2.c

```
int main(void)
{
    int choice, result, x, y;
    // 함수 포인터 배열을 선언하고 초기화한다.
    int (*pf[4])(int, int) = { add, sub, mul, div };

    while(1)
    {
        menu();

        printf("메뉴를 선택하시오:");
        scanf("%d", &choice);

        if( choice < 0 || choice >=4 )
            break;

        printf("2개의 정수를 입력하시오:");
        scanf("%d %d", &x, &y);

        result = pf[choice](x, y); // 함수 포인터를 이용한 함수 호출

        printf("연산 결과 = %d\n", result);
    }
    return 0;
}
```

함수 포인터 배열
선언

```
=====
0. 덧셈
1. 뺄셈
2. 곱셈
3. 나눗셈
4. 종료
=====
메뉴를 선택하시오:2
2개의 정수를 입력하시오:10 20
연산 결과 = 200
=====
0. 덧셈
1. 뺄셈
2. 곱셈
3. 나눗셈
4. 종료
=====
메뉴를 선택하시오:
```

fp2.c

```
int add(int x, int y)
{
    return x + y;
}

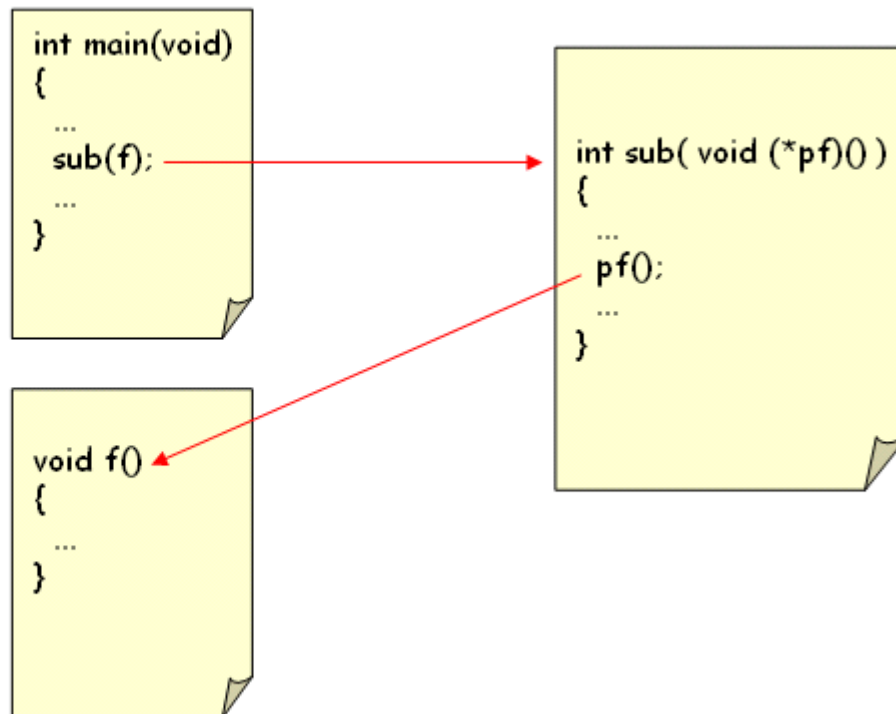
int sub(int x, int y)
{
    return x - y;
}

int mul(int x, int y)
{
    return x * y;
}

int div(int x, int y)
{
    return x / y;
}
```

함수 인수로서의 함수 포인터

- 함수 포인터도 인수로 전달이 가능하다.



fp2.c

```
#include <stdio.h>
#include <math.h>

double f1(double k);
double f2(double k);
double formula(double (*pf)(double), int n);

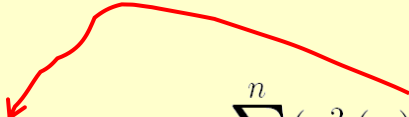
int main(void)
{
    printf("%f\n", formula(f1, 10));
    printf("%f\n", formula(f2, 10));
}

double formula(double (*pf)(double), int n)
{
    int i;
    double sum = 0.0;

    for(i = 1; i < n; i++)
        sum += pf(i) * pf(i) + pf(i) + 1; //
    return sum;
}
```

```
double f1(double k)
{
    return 1.0 / k;
}

double f2(double k)
{
    return cos(k);
}
```


$$\sum_1^n (f^2(k) + f(k) + 1)$$

```
13.368736
12.716152
```