

제14장 포인터 활용


유 준 범 (JUNBEOM YOO)

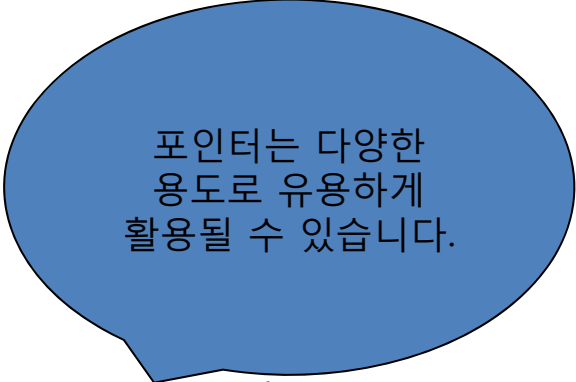
jbyoo@konkuk.ac.kr
<http://dslab.konkuk.ac.kr>

Ver. 2.0

※ 본 강의자료는 생능출판사의 "PPT 강의자료"를 기반으로 제작되었습니다.

이번 장에서 학습할 내용

- 
- 이중포인터란 무엇인가?
 - 포인터 배열
 - 함수 포인터
 - 다차원 배열과 포인터
 - void 포인터



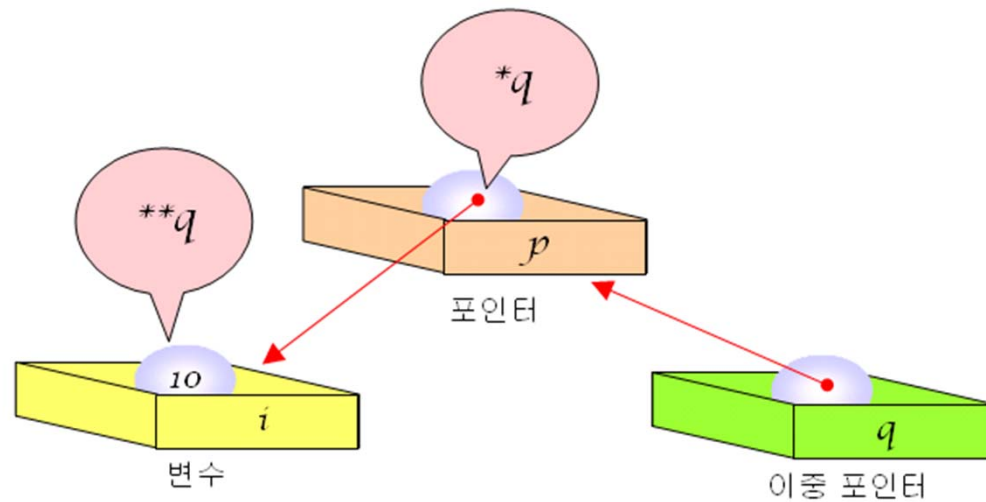
포인터는 다양한
용도로 유용하게
활용될 수 있습니다.



이중 포인터

- 이중 포인터(double pointer): 포인터를 가리키는 포인터

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



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;  
}
```

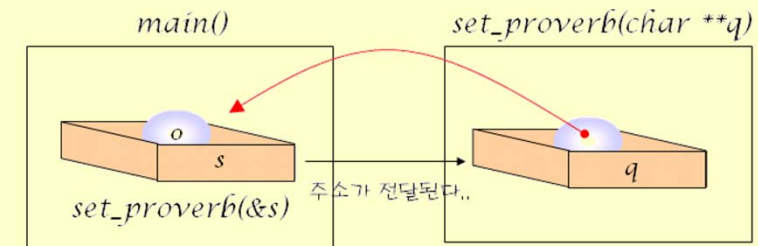
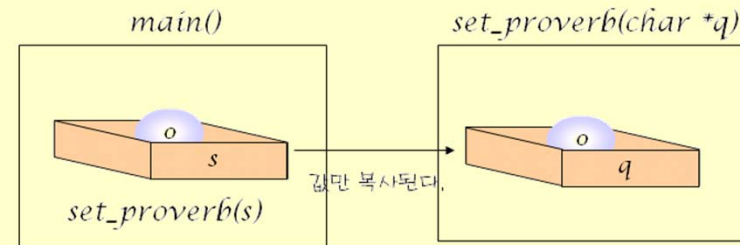
```
void set_proverb(char **q)
```

```
{
```

```
    static char *str1="A friend in need is a friend indeed";  
    static char *str2="A little knowledge is a dangerous thing";
```

```
    *q = str1;
```

```
}
```

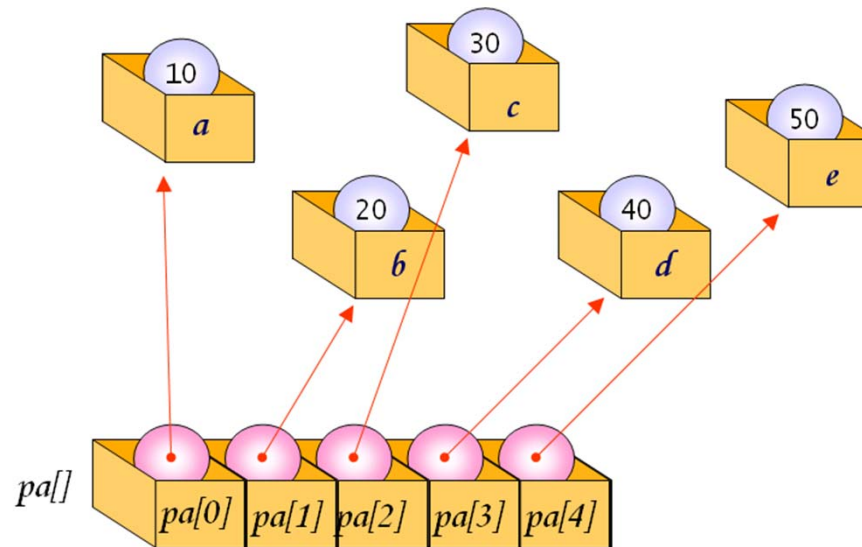


```
selected proverb = A friend in need is a friend indeed
```

포인터 배열

- 포인터 배열(array of pointers): 포인터를 모아서 배열로 만든 것

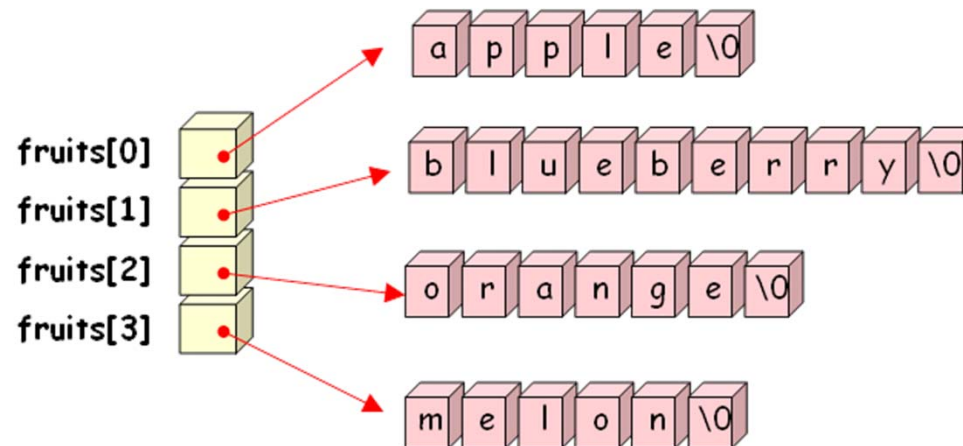
```
int a = 10, b = 20, c = 30, d = 40, e = 50;  
int *pa[5] = { &a, &b, &c, &d, &e };
```



문자열 배열

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

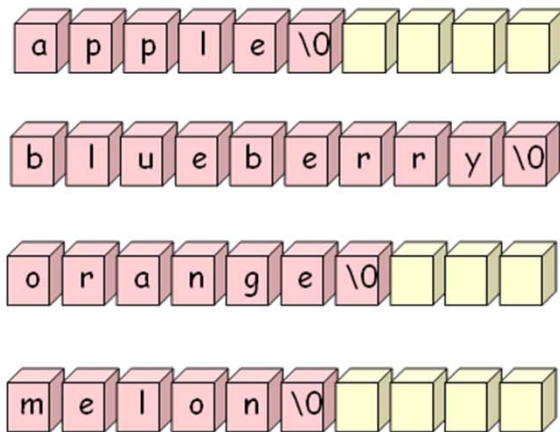
```
char *fruits[ ] = {  
    "apple",  
    "blueberry",  
    "orange",  
    "melon"  
};
```



문자열 배열 vs 2차원 배열

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

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



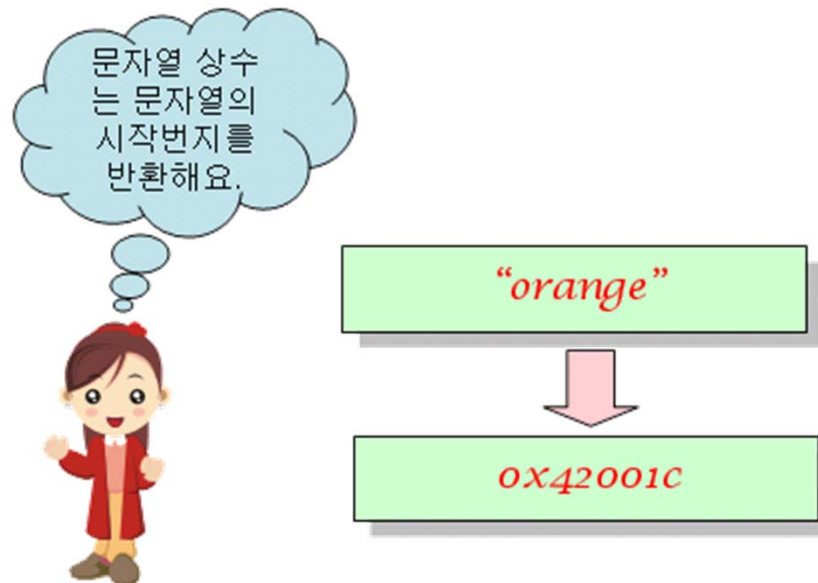
낭비되는
공간!

2차원 배열을 사
용하면 낭비되는
공간이 생성되죠.



문자열 상수의 의미

- 문자열 상수는 문자열의 시작 번지를 반환한다.
- 포인터 배열의 각 원소들은 이 시작 번지로 초기화된다.



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;
}
```

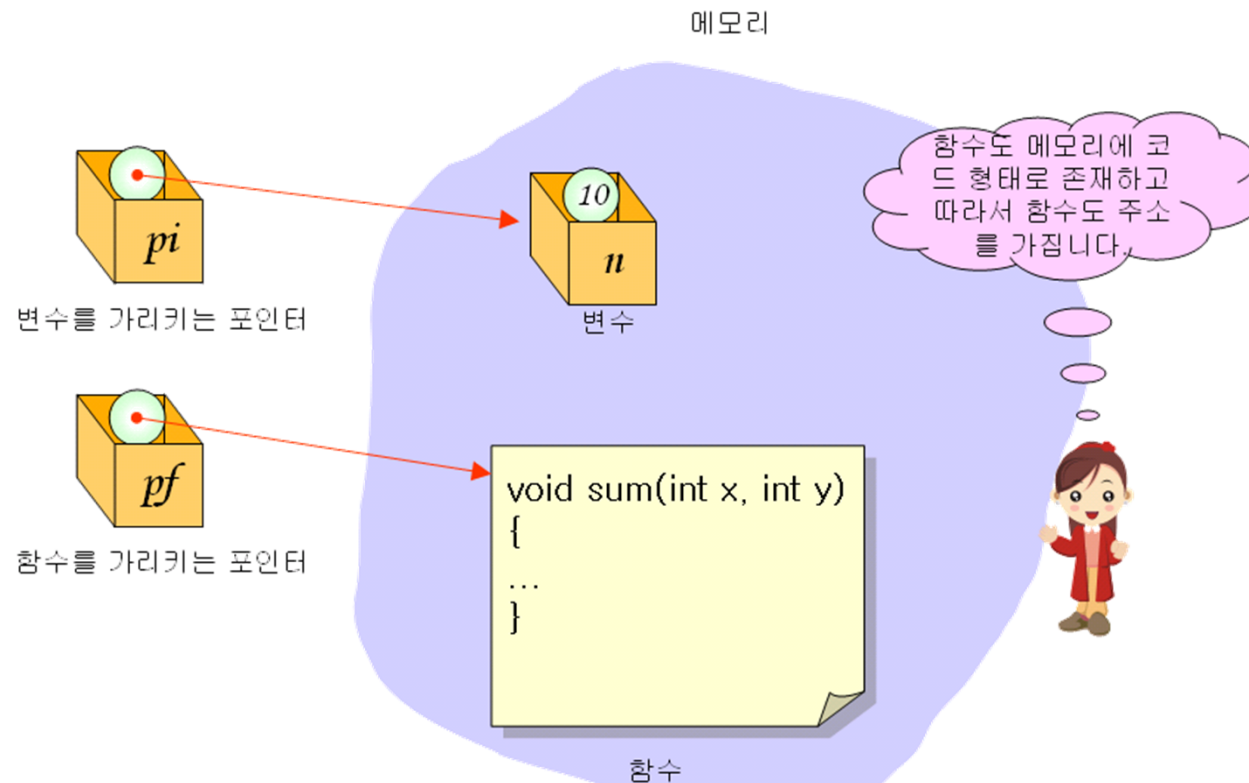


```
apple
blueberry
orange
melon
```

함수 포인터

- 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;                      // 함수 포인터에 함수 add()의 주소 대입
    result = pf(10, 20);           // 함수 포인터를 통한 함수 add() 호출
    printf("10+20은 %d\n", result);

    pf = sub;                      // 함수 포인터에 함수 sub()의 주소 대입
    result = pf(10, 20);           // 함수 포인터를 통한 함수 sub() 호출
    printf("10-20은 %d\n", result);

    return 0;
}
```

fp1.c

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

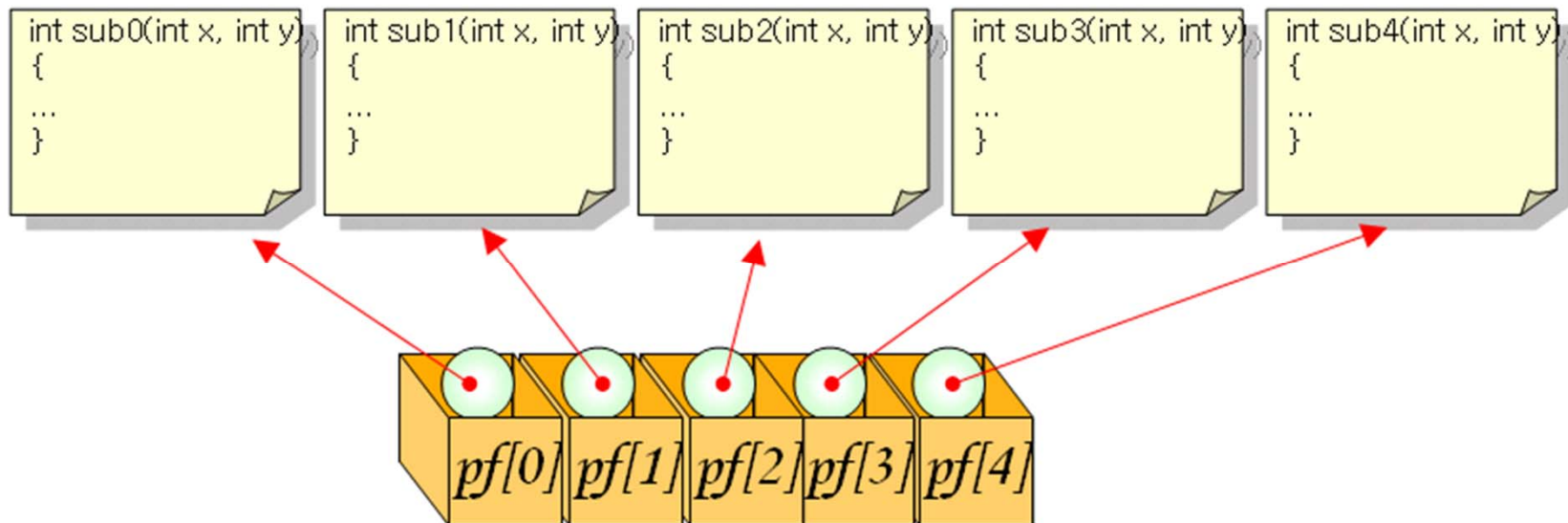
int sub(int x, int y)
{
    return x-y;
}
```



```
10+20 = 30
10-20 = -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 선택하십시오:");
        scanf("%d", &choice);

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

        printf("2개의 정수를 입력하십시오:");
        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

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;
}
```



- ```
=====
0. 덧셈
1. 뺄셈
2. 곱셈
3. 나눗셈
4. 종료
```

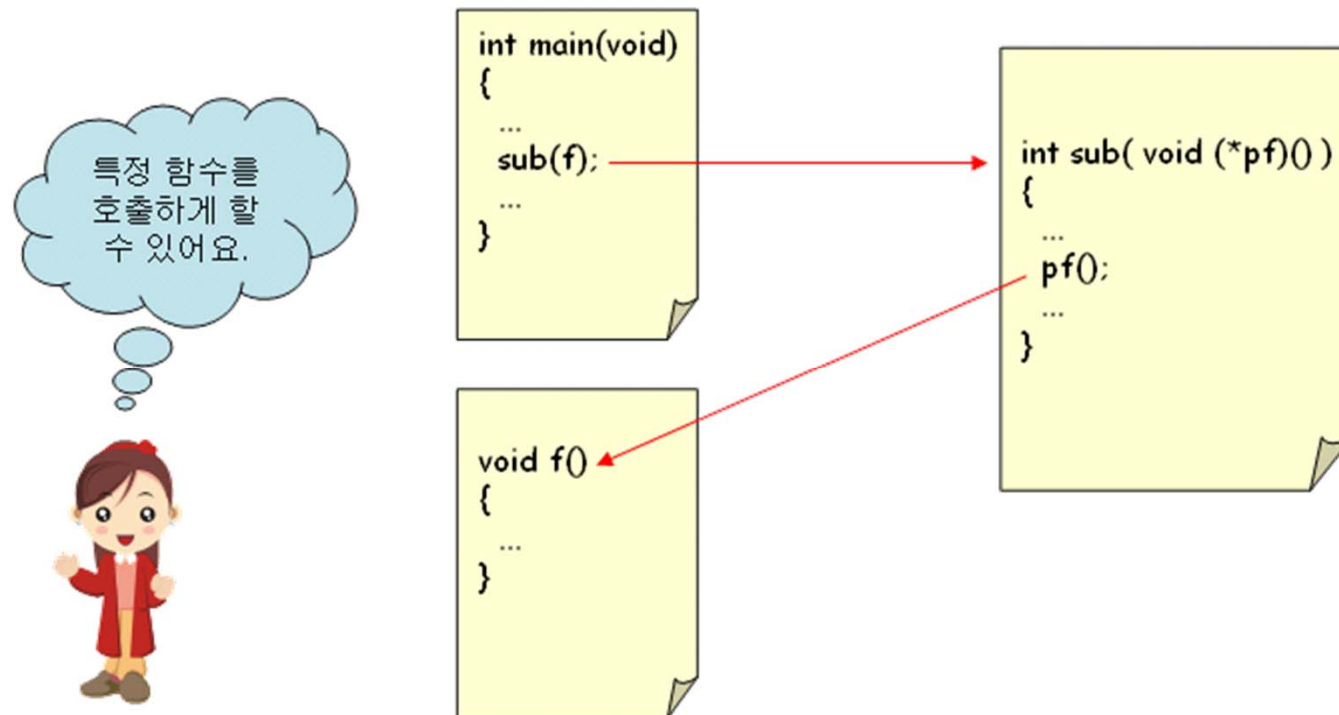
```
=====
메뉴를 선택하시오:2
2개의 정수를 입력하시오:10 20
연산 결과 = 200
```

- ```
=====
0. 덧셈
1. 뺄셈
2. 곱셈
3. 나눗셈
4. 종료
```

```
=====
메뉴를 선택하시오:
```

함수 인수로서의 함수 포인터

- 함수 포인터도 인수로 전달이 가능하다.



fp2.c



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

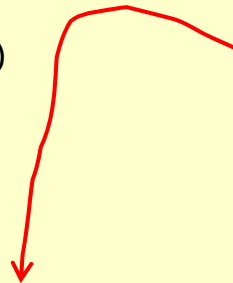
double f1(double k);
double f2(double k);
double formula(double (*pf)(double), int n);

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;
}
```

$$\sum_{1}^{n} (f^2(k) + f(k) + 1)$$



fp2.c

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

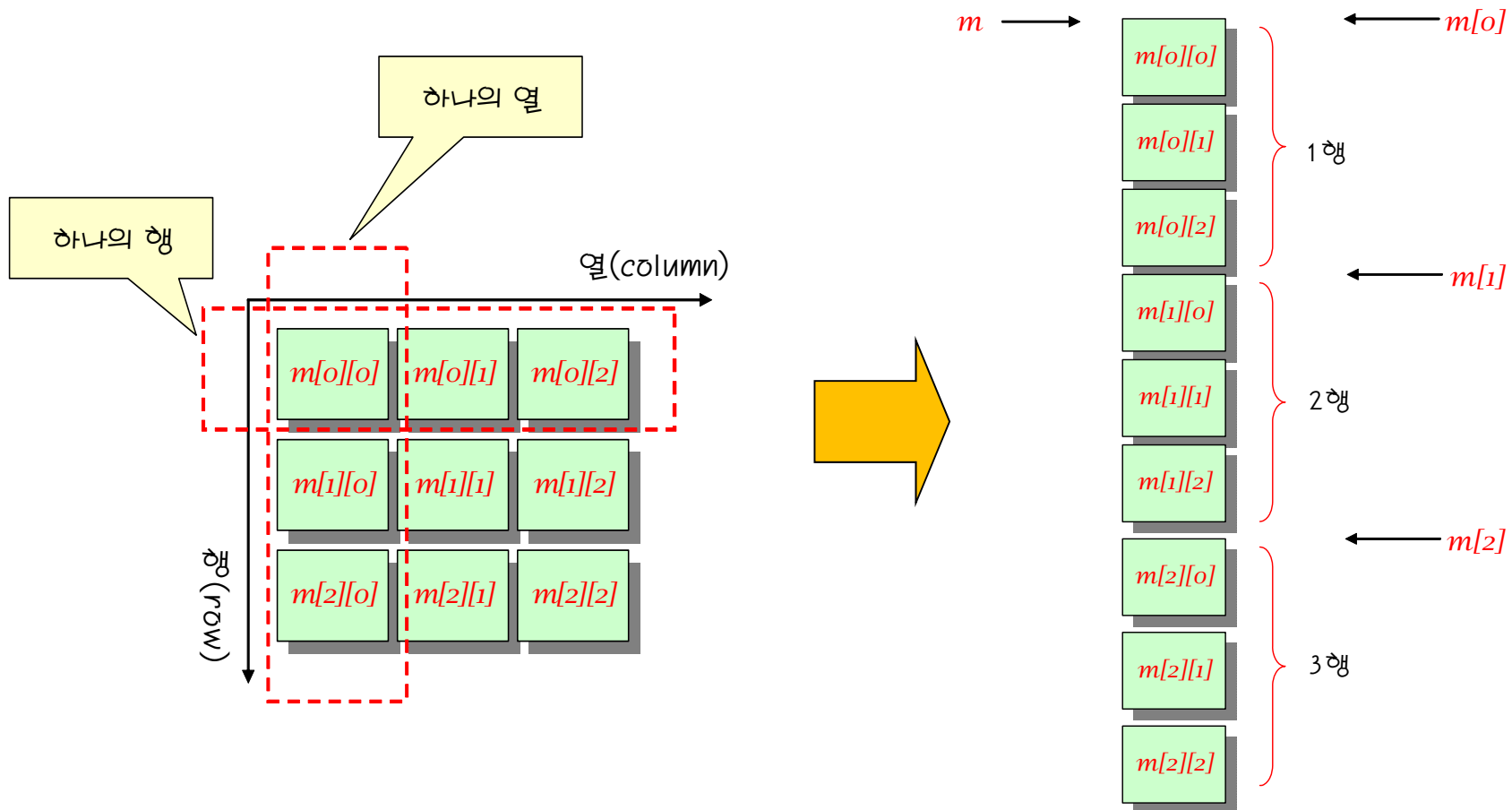
```
double f2(double k)
{
    return cos(k);
}
```



```
13.368736
12.716152
```

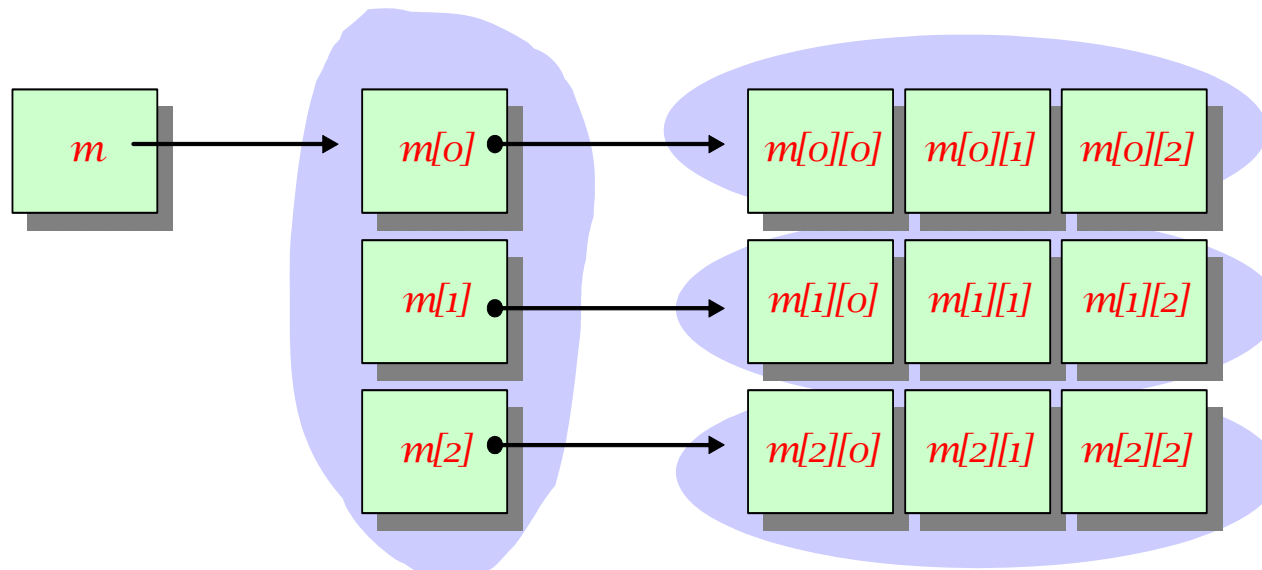
다차원 배열과 포인터

- 2차원 배열 `int m[3][3]`
- 1행->2행->3행->...순으로 메모리에 저장(행우선 방법)



2차원 배열과 포인터

- 배열 이름 m 은 $\&m[0][0]$
- $m[0]$ 는 1행의 시작 주소
- $m[1]$ 은 2행의 시작 주소



multi_array.c



```
// 다차원 배열과 포인터
#include <stdio.h>

int main(void)
{
    int m[3][3] = { 10, 20, 30, 40, 50, 60, 70, 80, 90 };

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

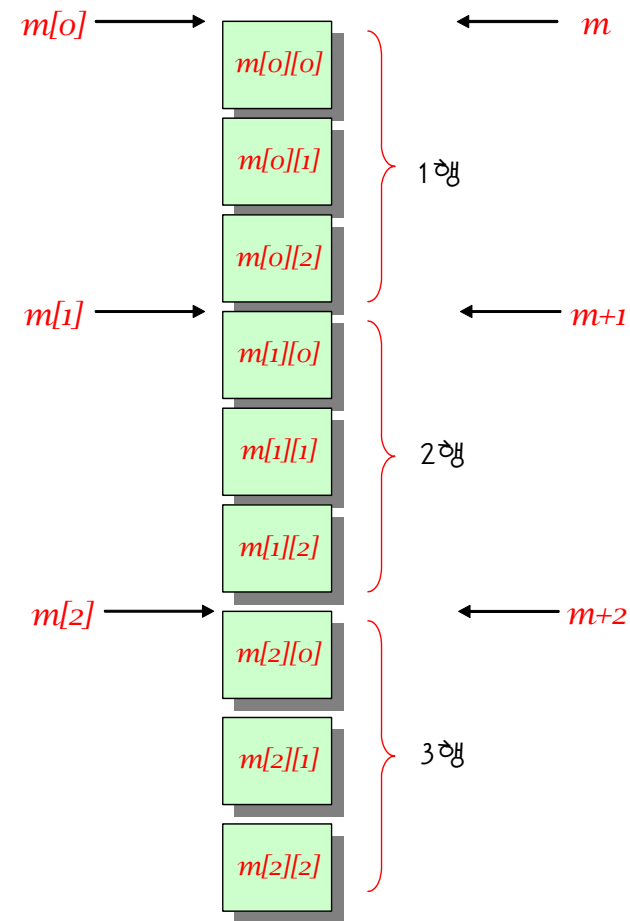
    return 0;
}
```



```
m = 1245020
m[0] = 1245020
m[1] = 1245032
m[2] = 1245044
&m[0][0] = 1245020
&m[1][0] = 1245032
&m[2][0] = 1245044
```

2차원 배열과 포인터 연산

- m 에 대한 연산의 의미
- m 은 $\&m[0][0]$
- $m+1$ 은 $m[1]$
- $m+2$ 은 $m[2]$



two_dim_array.c



```
#include <stdio.h>

int main(void)
{
    int m[3][3] = { 10, 20, 30, 40, 50, 60, 70, 80, 90 };

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

    return 0;
}
```



```
m = 1245020
m+1 = 1245032
m+2 = 1245044
m[0] = 1245020
m[1] = 1245032
m[2] = 1245044
```

포인터를 이용한 배열 원소 방문

- 행의 평균을 구하는 경우

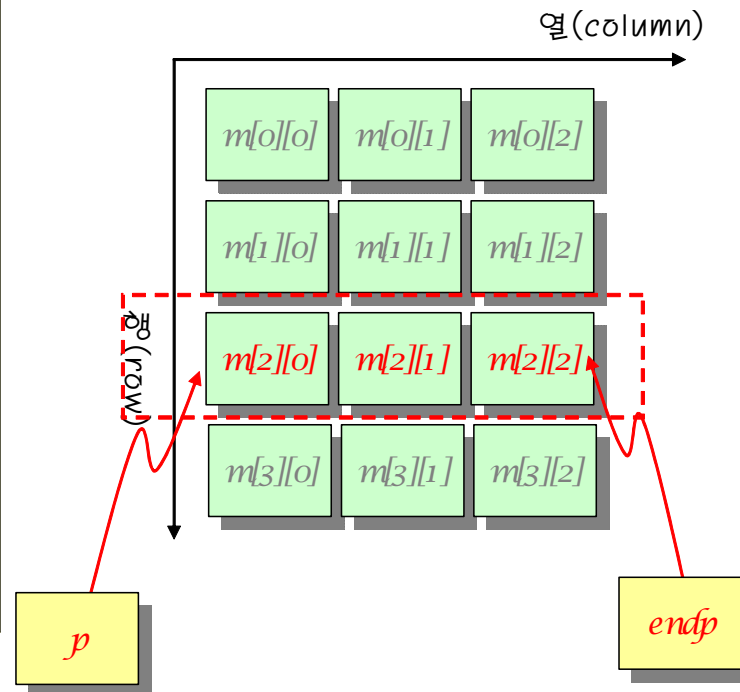
```
double get_row_avg(int m[][COLS], int r)
{
    int *p, *endp;
    double sum = 0.0;

    p = &m[r][0];
    endp = &m[r][COLS];

    while( p < endp )
        sum += *p++;

    sum /= COLS;

    return sum;
}
```



포인터를 이용한 배열 원소 방문

- 전체 원소의 평균을 구하는 경우

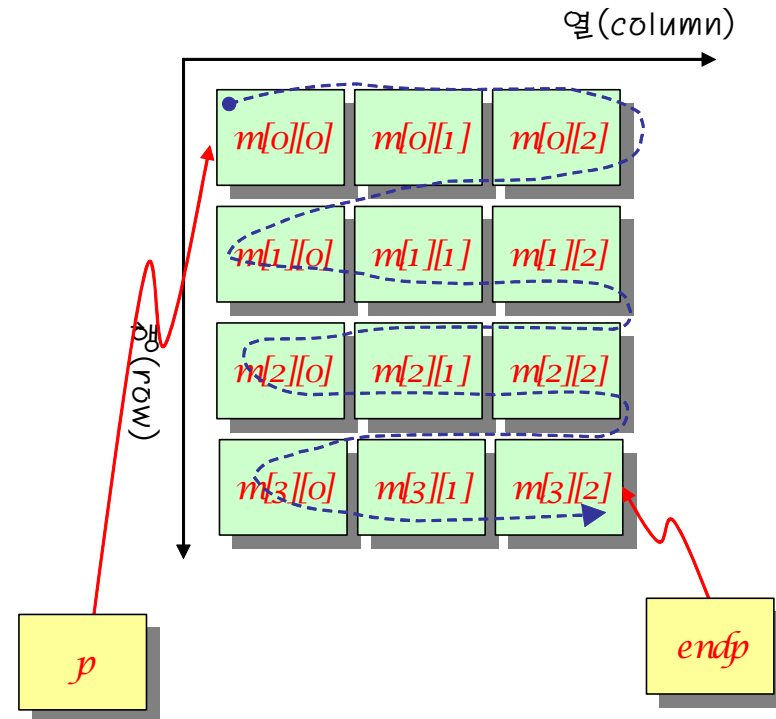
```
double get_total_avg(int m[][COLS])
{
    int *p, *endp;
    double sum = 0.0;

    p = &m[0][0];
    endp = &m[ROWS-1][COLS];

    while( p < endp )
        sum += *p++;

    sum /= ROWS * COLS;

    return sum;
}
```



void 포인터

- 순수하게 메모리의 주소만 가지고 있는 포인터
- 가리키는 대상물은 아직 정해지지 않음

(예) `void *vp;`

- 다음과 같은 연산은 모두 오류이다.

```
*vp;           // 오류
*(int *)vp;    // void형 포인터를 int형 포인터로 변환한다.
vp++;          // 오류
vp--;          // 오류
```

vp.c

```
#include <stdio.h>

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

    vp = a; // 가능
    vp = &a[2]; // 가능

    *vp = 35; // 오류
    vp++; // 오류

    *(int *)vp = 35; // 가능

    return 0;
}
```

main() 함수의 인수

- 지금까지의 main() 함수 형태

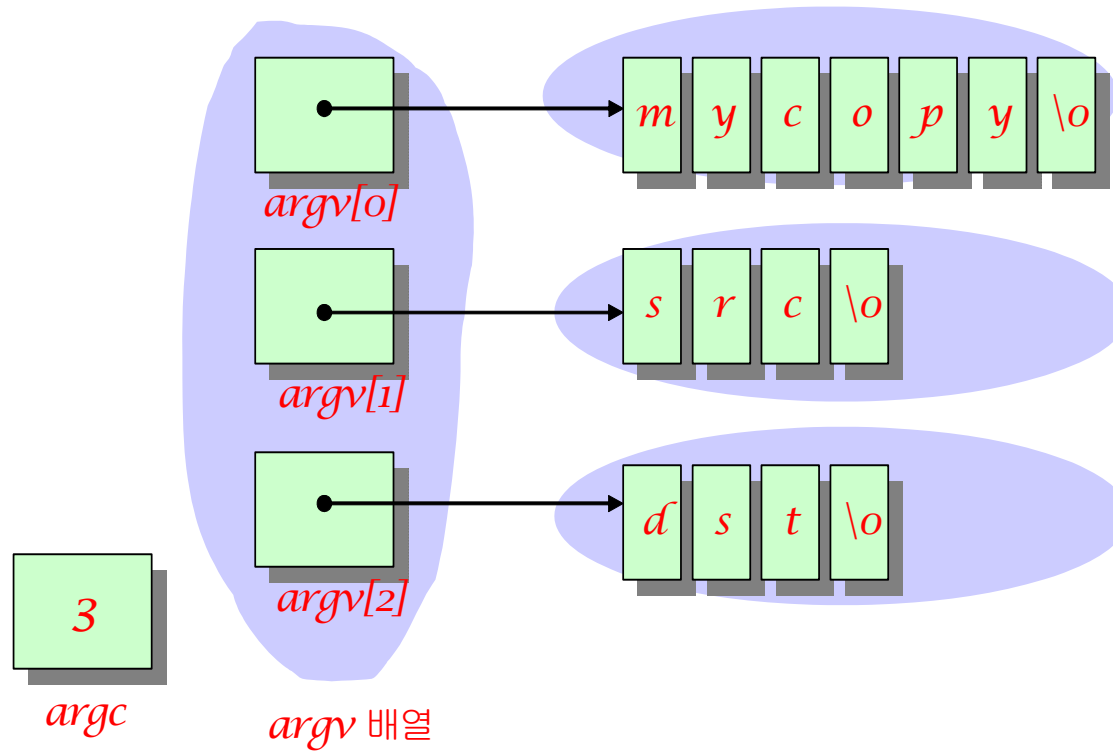
```
int main(void)
{
..
}
```

- 외부로부터 입력을 받는 main() 함수 형태

```
int main(int argc, char *argv[])
{
..
}
```

인수 전달 방법

```
C: \cprogram> mycopy src dst
```



main_arg.c



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

```
int main(int argc, char *argv[])
```

```
{
```

```
    int i = 0;
```

```
    for(i = 0; i < argc; i++)
```

```
        printf("명령어 라인에서 %d번째 문자열 = %s\n", i, argv[i]);
```

```
    return 0;
```

```
}
```



Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.

c:\cprogram\mainarg\Debug>mainarg src dst

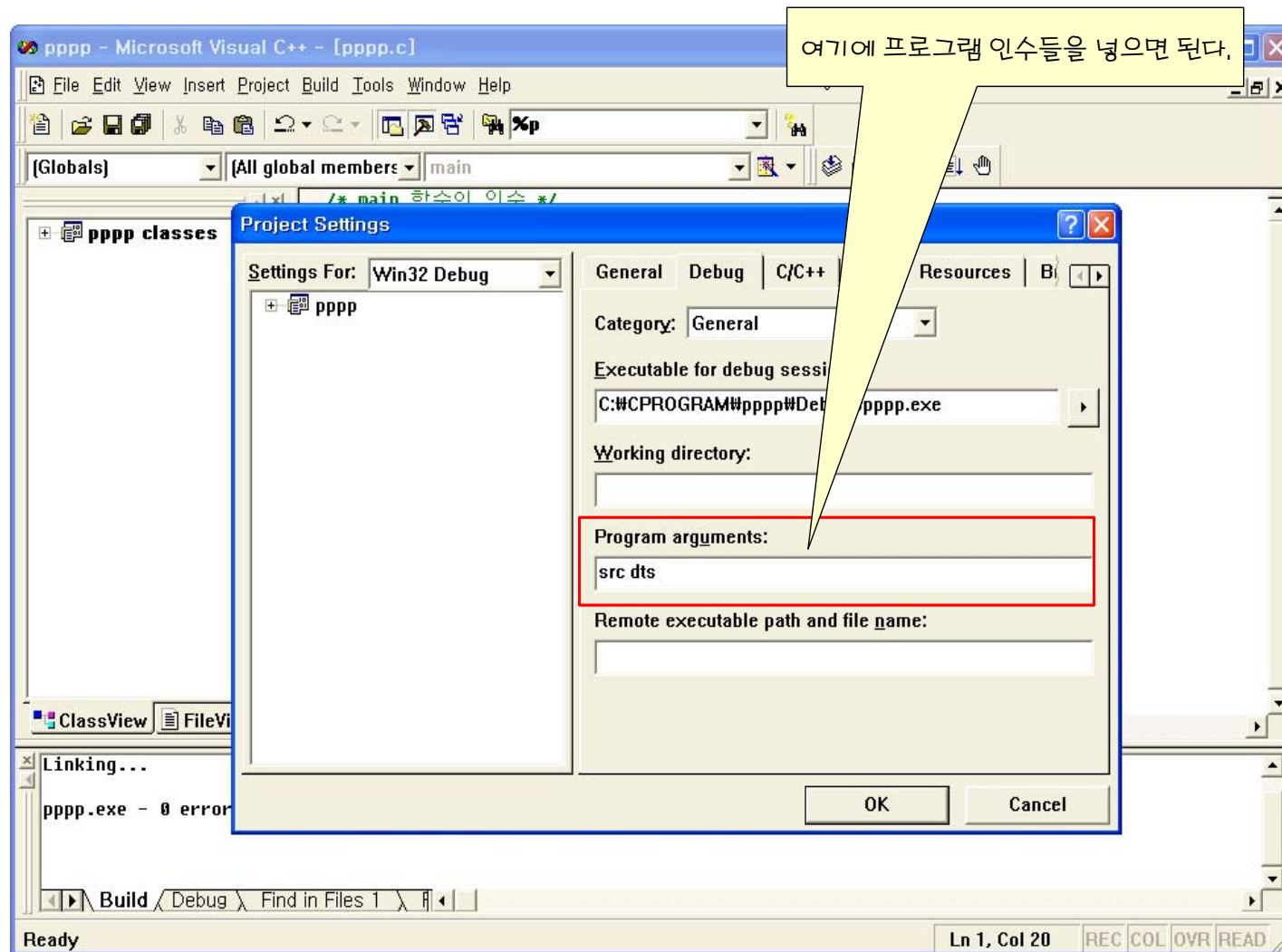
명령어 라인에서 0번째 문자열 = mainarg

명령어 라인에서 1번째 문자열 = src

명령어 라인에서 2번째 문자열 = dst

c:\cprogram\mainarg\Debug>

비주얼 C++ 프로그램 인수 입력 방법



mile2km.c



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

int main(int argc, char *argv[])
{
    double mile, km;

    if( argc != 2 ){
        printf("사용 방법: mile2km 거리\n");
        return 1;
    }
    mile = atof(argv[1]);
    km = 1.609 * mile;
    printf("입력된 거리는 %f km입니다. \n", km);

    return 0;
}
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\cprogram\mainarg\Debug>mainarg 10
입력된 거리는 16.090000 km입니다.
c:\cprogram\mainarg\Debug>
```

Q & A

