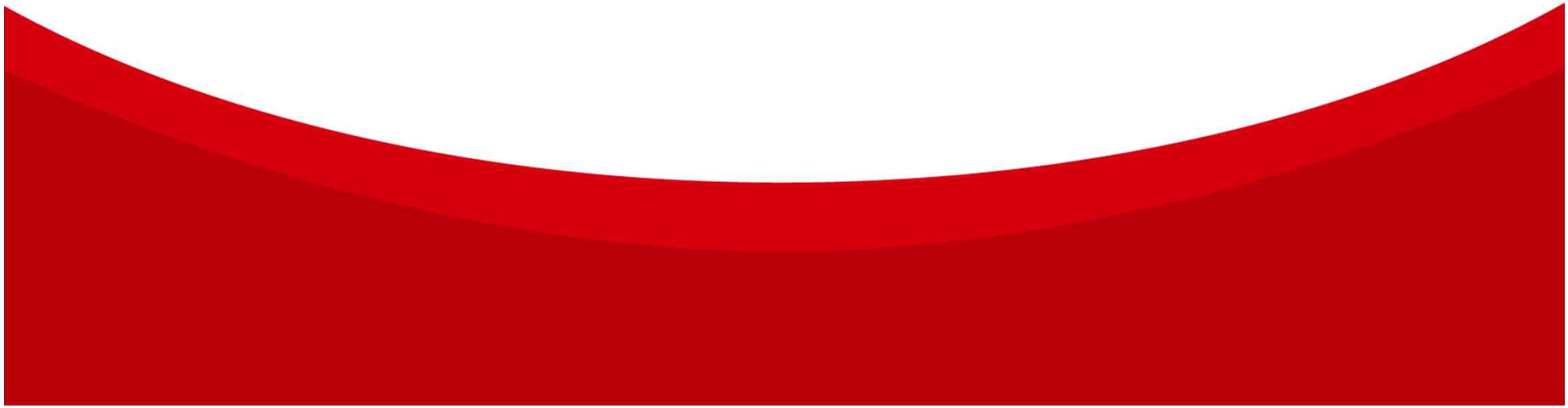
Chapter 12
함수 2-2



03 Question

두 정수를 입력받아 차를 구하고, 두 실수를 입력받아 차를 구하는 프로그램을 작성하시오.

입력 예

```
58 62
1.25 52.23
```

출력 예

```
두 정수의 차 : 4
두 실수의 차 : 50.980000
```

소스

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

int main()
{
	int a, b;
	double c, d;

	scanf("%d %d", &a, &b);
	scanf("%lf %lf", &c, &d);

	printf("두 정수의 차 : %d \n", abs(a - b));
	printf("두 실수의 차 : %f \n", fabs(c - d));

	return 0;
}
```

설명

- **#include <stdio.h>**

지금까지 사용자가 직접 함수를 작성하여 놓고 필요한 곳에서 호출하여 사용하는 방법을 알아보았다.

그런데 우리는 그 동안 printf, scanf와 같은 함수를 호출하여 사용하면서도 그 함수를 작성한 적이 없다.

이 함수는 C프로그램에 미리 정의되어 기본적으로 제공되는 함수이며 이러한 함수를 **표준라이브러리 함수**라고 한다.

표준라이브러리 함수를 저장해 놓은 파일을 헤더파일이라고 하며 printf와 같은 함수는 <stdio.h>라는 헤더파일에 포함되어 있다.

따라서 이 함수를 사용하기 위해 우리는 항상 위와 같이 프로그램에 <stdio.h> 파일을 포함시켰던 것이다.

- #include <stdlib.h>
  기본적인 표준 라이브러리(Standard library) 함수들을 모아놓은 헤더파일이다. abs() 함수를 사용하기 위해 포함시켰다.

- #include <math.h>
  여러 가지 수학함수를 정의해 놓은 헤더파일로 절대값(fabs), 제곱근(sqrt), 거듭제곱(pow), 올림(ceil), 버림(floor), 삼각비(sin, cos, tan), 로그(log) 등이 정의되어 있다.

- abs(a - b)
  abs는 ()안 정수의 절대값을 구하는 함수이다. <stdlib.h>에 포함되어 있다. Visual C++에서는 <math.h>에도 포함되어 있다.

- fabs(c - d)
  fabs는 ()안 실수의 절대값을 구하는 함수이다. <math.h>에 포함되어 있다.

자가진단 3

두 개의 정수를 입력받아 절대값이 더 큰 수를 출력하고, 두 개의 실수를 입력받아 절대값이 작은 수를 출력하는 프로그램을 작성하시오.

입력 예	출력 예
-50 40	-50
-12.34 5.67	5.67

04 Question

정사각형의 넓이를 입력받아서 한 변의 길이를 구하고, 밑과 지수를 입력받아 거듭제곱의 결과를 출력하는 프로그램을 작성하시오.

입력과 출력의 예

정사각형의 넓이 : 36
정사각형의 한 변의 길이 : 6.000000
밑과 지수 : 4 2
4.000000의 2.000000승은 16.000000입니다.

소스

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

int main()
{
	double area;	// 넓이
	double base;	// 밑
	double exp;	// 지수(exponent를 줄여서 나타냄)

	printf("정사각형의 넓이 : ");
	scanf("%lf", &area);
	printf("정사각형의 한 변의 길이 : %lf \n", sqrt(area));

	printf("밑과 지수 : ");
	scanf("%lf %lf", &base, &exp);
	printf("%lf의 %lf승은 %lf입니다. \n", base, exp, pow(base, exp));

	return 0;
}
```

설명

- sqrt(area)

sqrt는 제곱근을 구하는 함수로 sqrt(area)는 $\sqrt{area}$를 구하여 double형으로 리턴한다.

- pow(base, exp)

pow는 거듭제곱을 구하는 함수로 pow(base, exp)는 $base^{exp}$를 double형으로 리턴한다. area, base, exp에는 실수와 정수를 모두 사용할 수 있지만 함수 내에서는 double로 처리하여 리턴값은 항상 double임을 주의하자.

자가진단 4

원의 넓이를 입력받아 반지름의 길이를 소수 둘째자리까지 출력하는 프로그램을 작성하시오. (원의 넓이 = 반지름 * 반지름 * 3.14 식을 이용하시오)

입력 예	출력 예
314	10.00

- hint

원의 넓이 / 3.14 의 제곱근을 구하면 된다.

보충

$a * a = a^2$ 으로 표현할 수 있으며
$a^2 = b$의 식에서 b는 a의 제곱이라 하고, a는 b의 제곱근이라 한다.
제곱근은 루트(root)라고 하며 $\sqrt{\ }$ 기호로 표시한다.
위의식 $a^2 = b$라면 $a = \sqrt{b}$ 가 된다.
C언어의 루트함수는 sqrt이며 sqrt(b)라고 하면 $\sqrt{b}$를 나타내는 것이다.
정사각형의 넓이 = 한변의 길이 * 한변의 길이 이므로 sqrt(정사각형의 넓이) = 한변의 길이가 된다.

거듭제곱은 한 개의 값을 여러번 곱하는 연산을 의미하며
a^b 이란 a를 b번 곱하는 것을 의미한다.
5^3 이라고 표시하면 5 * 5 * 5를 나타내며 값은 125가 된다.
이때 곱해지는 수 a를 밑이라 하고 곱하는 횟수를 나타내는 b를 지수라 한다.

05 Question

반지름의 길이를 입력받아서 원의 넓이를 구하되, 소수 이하를 버림한 경우, 반올림한 경우, 올림한 경우를 각각 출력하는 프로그램을 작성하시오. (원주율은 3.14로 한다.)

입력 예 *원의 반지름* : 6

출력 예
원의 넓이
버림 : 113
반올림 : 113
올림 : 114

소스

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

double round(double val)
{
	return floor(val + 0.5);
}

int main()
{
	double radius, area, pie = 3.14;

	printf("원의 반지름 : ");
	scanf("%lf", &radius);

	area = pow(radius, 2) * pie;

	printf("원의 넓이\n");
	printf("버림 : %.0f \n", floor(area));
	printf("반올림 : %.0f \n", round(area));
	printf("올림 : %.0f \n", ceil(area));

	return 0;
}
```

설명

- floor(area)
  floor()함수는 ()안 실수의 소수점 이하를 버림한 결과를 리턴한다. 여기에서 버림이란 주어진 값보다 크지 않은 최대의 정수 값을 의미한다. 예를 들어 floor(-2.5)는 -2가 아니라 -3이 된다는 것이다. (-2는 -2.5보다 큰 수이므로)

- ceil(area)
  ceil()함수는 ()안 실수의 소수점 이하를 올림한 결과를 리턴한다. 여기에서 올림이란 주어진 값보다 작지 않은 최소의 정수 값을 의미한다. 다음의 표는 floor와 ceil함수의 사용 결과에 대한 예시이다.

사용예	결과	사용예	결과
floor(1.0)	1	ceil(1.0)	1
floor(1.1)	1	ceil(1.1)	2
floor(1.9)	1	ceil(1.9)	2
floor(2.5)	2	ceil(2.5)	3
floor(-2.5)	-3	ceil(-2.5)	-2
floor(-3.0)	-3	ceil(-3.0)	-3

- double round(double val)
  {
      return floor(val+0.5);
  }
  실수값을 전달받아 소수 첫째 자리에서 반올림한 정수값을 리턴하는 사용자정의 함수이다. <math.h>에 버림과 올림함수는 정의되어 있으나 반올림함수는 정의되어 있지 않다. 따라서 위와 같이 어떤 실수에 0.5를 더한 후 버림을 한 값을 리턴하게 되면 반올림한 결과 값이 된다.

자가진단 5

세 개의 실수를 입력받아 가장 큰 수를 올림한 정수를 출력하고 가장 작은 수를 버림한 정수를 출력한 후 남은 수를 반올림한 정수를 출력하는 프로그램을 작성하시오.
입력되는 실수는 −1000 이상 1000 이하이다.

입력 예

3.45 51.48 -100.1

출력 예

52 -101 3

06 Question

반지름의 길이를 입력받아서 원의 둘레의 길이를 구하여 출력하는 프로그램을 작성하시오. (원주율은 3.14로 하고 반올림하여 소수 둘째자리까지 출력한다.)

입력 예 5.5

출력 예 34.54

소스1

```
#include <stdio.h>

int main()
{
	double r, ci;	// 반지름(radius), 원주(circumference)
	const double PI = 3.14;
	scanf("%lf", &r);
	ci = r * 2 * PI;
	printf("%.2f\n", ci);

	return 0;
}
```

설명

- const double PI = 3.14;
  double형 **상수 PI를 선언**하고 3.14를 대입한다. 상수란 변경이 불가능한 고정된 값을 말한다. 변수를 선언할 때 앞에 const를 붙여주면 이는 이 변수에 값을 넣어서 상수로 사용하겠다는 의미가 되며 따라서 **이 때 선언된 변수는 더 이상 값을 변경시킬 수 없는 상수**가 된다.

  이렇게 const로 상수 자료형을 선언할 때는 선언과 동시에 값을 대입해 주어야 하며 프로그램 중간에 값을 변경시킬 수 없다. 만약 const로 선언된 값을 변경시키는 코드가 프로그램 내에 존재하면 컴파일 에러가 발생한다. 즉, 위의 문장을

  ```
  const double PI;
  PI = 3.14;
  ```

  이렇게 작성을 하고 컴파일을 하려고 하면 에러가 발생된다. 이러한 변수형 상수는 **값이 고정되어 있어야 하는 경우 프로그램 작성 중간에 값을 변경하는 실수를 방지하고자 주로 사용된다.**

소스2

```
#include <stdio.h>
#define PI 3.14
#define SIK r * 2 * PI

int main()
{
        double r, ci;    // 반지름, 원주
        scanf("%lf", &r);
        ci = SIK;
        printf("%.2lf\n", ci);

        return 0;
}
```

설명

- #define PI 3.14
  컴파일을 할 때 프로그램 내에 존재하는 모든 PI 를 3.14로 변환한다. '#'로 시작되는 문장을 전처리기 지시자라고 하며 그동안 늘 써왔던 #include<헤더파일이름>은 컴파일할 때 <>안의 헤더파일을 이 프로그램에 포함을 시키라는 의미이다.

- 반면에
  #define 첫 번째 두 번째와 같은 문장은 이 프로그램 내에서 첫 번째와 같은 것을 모두 찾아서 두 번째와 같이 치환하라는 것이다. 여기에서 첫 번째 영역을 매크로라고 한다. 소스1과 결과는 같지만 소스1에서는 PI라는 변수명을 사용하여 메모리를 확보하고 그 위치에 상수 3.14를 저장하여 사용하는 것이지만, 매크로를 사용하게 되면 메모리가 필요하지 않고 프로그램에 직접 값을 입력한 것과 같게 되는 것이다.

- ci = SIK;
  SIK은 매크로이므로 r * 2 * PI로 치환된다.
  그런데 PI 역시 매크로이므로 3.14로 치환이 되어 이 문장은 최종적으로
  ci = r * 2 * 3.14;로 변환이 되어 컴파일 된다.

자가진단 6

main 함수 내에는 숫자를 사용하지 말고 1, 2, 3 세 개의 숫자를 조합하여 가능한 한 모든 합을 출력하는 프로그램을 작성하시오.

출력 예

```
1 + 1 = 2
1 + 2 = 3
1 + 3 = 4
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
3 + 1 = 4
3 + 2 = 5
3 + 3 = 6
```

- hint

#define one 1....

01번

6 개의 정수를 입력 받아 배열에 저장하고
오름차순으로 정렬하고
출력하는 프로그램을 작성하시오

```
int main() {

    int a[6];
    for (int i=0;i<6;i++)
        scanf("%d", &a[i]);

    for (int i=0;i<5;i++) {
        for (int j=i+1; j<6;j++) {
            if (a[i] > a[j]) {  //swap
                int tmp = a[i];
                a[i] = a[j];
                a[j] = tmp;
            }
        }
    }
    for (int i=0;i<6;i++)
        printf("%d ", a[i]);
}
```

```
int main() {
    int a[6];

    for (int i=0;i<6;i++)
        scanf("%d", &a[i]);                    input

    for (int i=0;i<5;i++) {
        for (int j=i+1; j<6;j++) {
            if (a[i] > a[j]) {  //swap
                int tmp = a[i];
                a[i] = a[j];                   sort
                a[j] = tmp;
            }
        }
    }

    for (int i=0;i<6;i++)
        printf("%d ", a[i]);                   output
}
```

=>구조화(=함수로 나누기)

```
int main() {

    int a[6];

    input(~~);
    sort(~~);
    output(~~);

}
```

```
int main() {

    int a[6];
    input(a, 6);
    sort(a,6);
    output(a,6);
}
```

```
void input(int a[], int cnt) {
    for (int i=0;i<cnt;i++)
        scanf("%d", &a[i]);
}
void sort(int a[], int cnt) {
    for (int i=0;i<cnt-1;i++) {
        for (int j=i+1; j<cnt;j++) {
            if (a[i] > a[j]) {  //swap
                int tmp = a[i];
                a[i] = a[j];
                a[j] = tmp;
            }
        }
    }
}
void output(int a[], int cnt) {
    for (int i=0;i<cnt;i++)
        printf("%d ", a[i]);
}
```

```
for (int i=0;i<6;i++)
    scanf("%d", &a[i]);
```

```
for (int i=0;i<5;i++) {
    for (int j=i+1; j<6;j++) {
        if (a[i] > a[j]) {  //swap
            int tmp = a[i];
            a[i] = a[j];
            a[j] = tmp;
        }
    }
}
```

```
for (int i=0;i<6;i++)
    printf("%d ", a[i]);
```

자가진단 1

10 이하의 자연수 n을 입력받고 n개의 정수를 입력받아 내림차순으로 정렬하여 출력하는 프로그램을 작성하시오. (배열을 전달하는 함수를 이용한다.)

입력 예

```
4
10 9 2 15
```

출력 예

```
15 10 9 2
```

07 Question

5개의 정수를 입력받아 오름차순으로 정렬하여 출력하는 프로그램을 작성하시오.

입력 예 9 2 6 8 3

출력 예 2 3 6 8 9

소스

```
#include <stdio.h>
#define N 5
#define SWAP(x, y) {int z = x; x = y; y = z;}

void input(int a[], int cnt)
{
        int i;
        for (i = 0; i < cnt; i++) {
                scanf("%d", &a[i]);
        }
}
```

```
void sort(int a[], int cnt)
{
        int i, j;
        for (i = cnt - 1; i > 0; i--) {
                for (j = 0; j < i; j++) {
                        if (a[j] > a[j + 1]) {
                                SWAP(a[j], a[j + 1]);
                        }
                }
        }
}

void output(int a[])
{
        int i;
        for (i = 0; i < N; i++) {
                printf("%d ", a[i]);
        }
}

int main()
{
```

```
    int arr[N];

    input(arr, N);
    sort(arr, N);
    output(arr);

    return 0;
}
```

- #define SWAP(x, y) {int z = x; x = y; y = z;}
  매크로에서 ()는 일반적인 함수의 ()처럼 매개변수와 같은 역할을 한다. 이렇게 함수처럼 사용되는 것을 매크로 함수라고 한다. 위 sort함수 내에서 사용된 코드가 어떻게 치환되어 컴파일 되는지를 살펴보자.

  SWAP(a[j], a[j + 1]); → {int z = a[j]; a[j] = a[j + 1]; a[j + 1] = z;}

  매크로 함수에서 x라고 된 부분은 모두 a[j]로, y라고 된 부분은 모두 a[j + 1]로 치환되는 것을 알 수 있다. 이러한 매크로 함수가 일반적인 함수와 다른 점은 프로그램 **실행시에 함수로 작동되는 것이 아니라 컴파일 시에 코드 자체를 바꾸어 주는 것이므로 전달하는 과정에서 자료형을 정의할 필요가 없고 실행 속도도 빠르다는 것이다**. 하지만 내용이 복잡해지면 코드의 크기가 지나치게 커질 수 있으며 실수에 의한 에러가 발생할 가능성이 높아지기 때문에 주로 간단한 코드에 한해 사용하는 것이 좋다.

- input(arr, N);
  output(arr);

  함수를 호출하면서 매크로 상수를 이용해 자료의 개수를 전달하는 방법과 자료의 개수를 전달하지 않고 함수 내에서 매크로 상수를 직접 사용하는 방법 등 다양한 활용 방법을 보여준다.

- **버블정렬**
  버블정렬은 바로 옆의 자료와 비교해 가면서 가장 큰 수를 맨 끝(i위치)에 오도록 하는 작업을 반복해 나가는 방법이다. 위 코드에서 첫 번째 for문의 i가 최대값이 와야할 위치이고 j값은 바로 뒤 자료와 비교할 위치를 나타낸다. 위 자료를 버블정렬로 오름차순 정렬하는 과정은 다음과 같다.

a[0]	a[1]	a[2]	a[3]	a[4]
9	2	6	8	3

원래 주어진 수

- 1단계 : 왼쪽부터 차례대로 오른쪽과 비교하여 가장 큰 수가 i(a[4])에 오도록 한다.

a[0]	a[1]	a[2]	a[3]	a[4]
9	2	6	8	3

a[0]과 a[1]을 비교한다.
a[1]이 더 작으므로
a[0]과 교환한다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	9	6	8	3

a[1]과 a[2]를 비교한다.
a[2]가 더 작으므로
a[1]과 교환한다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	6	9	8	3

a[2]와 a[3]을 비교한다.
a[3]이 더 작으므로
a[2]와 교환한다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	6	8	9	3

a[3]과 a[4]를 비교한다.
a[4]가 더 작으므로
a[3]과 교환한다.

- 2단계 : 그 다음으로 큰 수가 i(a[3])에 오도록 처음부터 다시 비교하여 교환한다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	6	8	3	9

a[0]과 a[1]을 비교한다.
a[1]이 더 크므로
교환하지 않는다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	6	8	3	9

a[1]과 a[2]를 비교한다.
a[2]가 더 크므로
교환하지 않는다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	6	8	3	9

a[2]와 a[3]을 비교한다.
a[3]이 더 작으므로
a[2]와 교환한다.

- 3단계 : 그 다음으로 큰 수가 i(a[2])에 오도록 처음부터 다시 비교하여 교환한다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	6	3	8	9

a[0]과 a[1]을 비교한다.
a[1]이 더 크므로
교환하지 않는다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	6	3	8	9

a[1]과 a[2]를 비교한다.
a[2]가 더 작으므로
a[1]과 교환한다.

- 4단계 : 그 다음으로 큰 수가 i(a[1])에 오도록 처음부터 다시 비교하여 교환한다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	3	6	8	9

a[0]과 a[1]을 비교한다.
a[1]이 더 크므로
교환하지 않는다.

- 5단계 : a[0]만 남았으므로 더 이상 진행하지 않는다. 버블정렬이 종료되었다.

a[0]	a[1]	a[2]	a[3]	a[4]
2	3	6	8	9

오름차순 버블정렬 종료

- 데이터 5개에 대한 버블정렬 역시 선택정렬과 마찬가지로 모두 4단계를 거침을 알 수 있다. 일반적으로 데이터 N개에 대한 버블정렬은 모두 N-1단계를 거친다. 또한 I 단계에서 a[i]에 최소값이 오도록 N-i번의 비교가 발생된다.

자가진단 7

10개의 정수를 입력받아 버블정렬로 내림차순 정렬을 하면서 하나의 단계가 끝날 때마다 그 정렬 결과를 출력하는 프로그램을 작성하시오.

입력 예

15 93 26 8 43 10 25 88 75 19

출력 예

93 26 15 43 10 25 88 75 19 8
93 26 43 15 25 88 75 19 10 8
93 43 26 25 88 75 19 15 10 8
93 43 26 88 75 25 19 15 10 8
93 43 88 75 26 25 19 15 10 8
93 88 75 43 26 25 19 15 10 8
93 88 75 43 26 25 19 15 10 8
93 88 75 43 26 25 19 15 10 8
93 88 75 43 26 25 19 15 10 8

08 Question

두 개의 정수 a, b를 입력받아 a보다 10 큰 수와 b보다 5 작은 수의 곱을 구하여 출력하는 프로그램을 작성하시오.

입력 예 10 20

출력 예 (10 + 10) * (20 - 5) = 300

소스

```
#include <stdio.h>
#define MULTI(x, y) (x) * (y)

int main()
{
        int a, b, c;
        scanf("%d %d", &a, &b);
        c = MULTI(a + 10, b - 5);
        printf("(%d + 10) * (%d - 5) = %d\n", a, b, c);
        return 0;
}
```

설명

- #define MULTI(x, y) (x) * (y)
  두 개의 자료를 받아서 그 곱을 연산하는 매크로 함수이다.
  두 자료 x와 y에 각각 ()를 붙여준 것에 유의해야 한다.
  ()를 붙였을 때와 생략했을 때 각각 MULTI(a + 10, b - 5);를 치환한 결과값을 살펴보자.

  (x) * (y) → (a + 10) * (b - 5)
  x * y → a + 10 * b - 5

  ()를 생략하게 되면 먼저 10 * b의 연산을 하게 되므로 전혀 다른 결과가 나오는 것을 확인할 수 있다. 따라서 매크로 함수를 사용할 때는 가능한 모든 매개변수에 ()를 써 주는 것이 안전하다.

자가진단 8

정수 두 개를 입력받아서 매크로 함수를 작성하여 두 수의 차를 제곱한 값과 합을 세제곱한 값을 각각 출력하는 프로그램을 작성하시오. (출력시 거듭제곱은 '^'로 표시하기로 한다.)

입력 예

```
5 10
```

출력 예

```
(5 - 10) ^ 2 = 25
(5 + 10) ^ 3 = 3375
```

Thank You!!!