Introduction to Computer and Lab

Homework #4

Due date: Apr 28, 2016

학번: 201404051

이름: 정 용 석

1. 전체 구구단을 출력

1.1 Solution

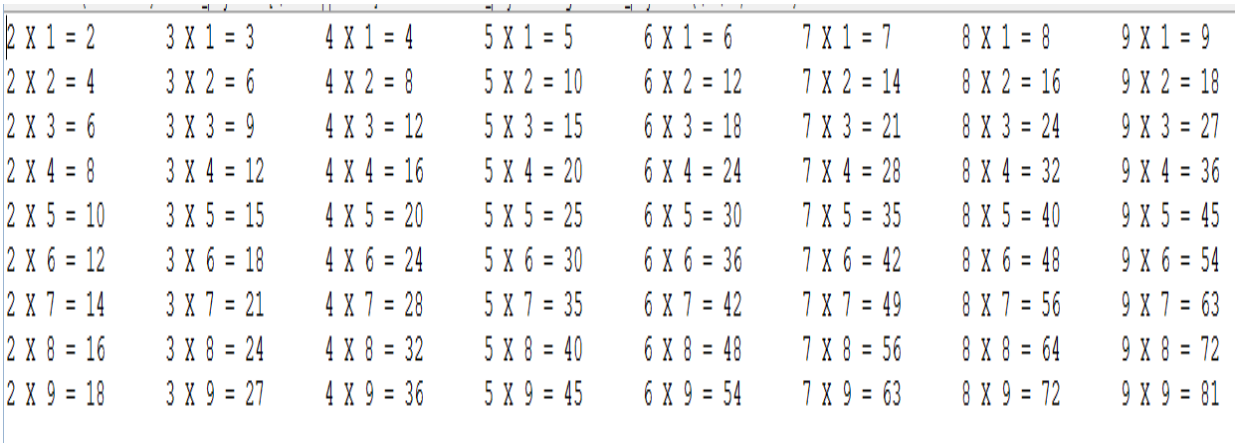
해결 방법은 굉장히 간단하다. 반복 문을 2 개를 사용해야 하는 것은 당연하고, 첫 번째와 두 번째 반복 문을 어떤 식으로 만드느냐가 핵심이다. 일단 주어진 출력 문을 보면 옆으로는 구구단의 단수가 높아지고, 밑으로는 각 단수의 2 에서 9 까지 곱한 값을 출력한다. 따라서 가로의 영역을 담당하는 두 번째 반복 문에 단수를 넣고 곱할 값을 첫 번째 반복 문에 넣는 것이 맞다. 따라서 첫 번째 반복 문에는 각 단수를 곱할 1 에서 9 까지의 수를 반복하게 넣고, 두 번째 반복 문에는 2~9 까지 단수를 넣어 주면 마무리된다.

1.2 Source Code

```
void gugudan()   // 구구단 출력
{
     int x, y; //x는 앞의 수, y는 뒤의 수

     for (y = 1; y <= 9; y++)
     {
          for (x = 2; x <= 9; x++) //2단~9단
               printf("%d X %d = %d\t", x, y, x * y);
          printf("\n");
     }
}
```

1.3 Result (Snap Shot)



2 X 1 = 2	3 X 1 = 3	4 X 1 = 4	5 X 1 = 5	6 X 1 = 6	7 X 1 = 7	8 X 1 = 8	9 X 1 = 9
2 X 2 = 4	3 X 2 = 6	4 X 2 = 8	5 X 2 = 10	6 X 2 = 12	7 X 2 = 14	8 X 2 = 16	9 X 2 = 18
2 X 3 = 6	3 X 3 = 9	4 X 3 = 12	5 X 3 = 15	6 X 3 = 18	7 X 3 = 21	8 X 3 = 24	9 X 3 = 27
2 X 4 = 8	3 X 4 = 12	4 X 4 = 16	5 X 4 = 20	6 X 4 = 24	7 X 4 = 28	8 X 4 = 32	9 X 4 = 36
2 X 5 = 10	3 X 5 = 15	4 X 5 = 20	5 X 5 = 25	6 X 5 = 30	7 X 5 = 35	8 X 5 = 40	9 X 5 = 45
2 X 6 = 12	3 X 6 = 18	4 X 6 = 24	5 X 6 = 30	6 X 6 = 36	7 X 6 = 42	8 X 6 = 48	9 X 6 = 54
2 X 7 = 14	3 X 7 = 21	4 X 7 = 28	5 X 7 = 35	6 X 7 = 42	7 X 7 = 49	8 X 7 = 56	9 X 7 = 63
2 X 8 = 16	3 X 8 = 24	4 X 8 = 32	5 X 8 = 40	6 X 8 = 48	7 X 8 = 56	8 X 8 = 64	9 X 8 = 72
2 X 9 = 18	3 X 9 = 27	4 X 9 = 36	5 X 9 = 45	6 X 9 = 54	7 X 9 = 63	8 X 9 = 72	9 X 9 = 81

2. 직각 삼각형 출력하기

2.1 Solution

밑변과 높이가 N 인 직각 삼각형이다. 하지만 삼각형의 모양이 왼쪽 하단을 직각으로 하는 삼각형이다. 따라서 줄이 바뀔수록 *의 개수가 증가하는 모습을 보인다. 반복 문의 첫 번째는 단순히 줄을 띄우는 역할을 하고, 두 번째 반복 문이 *을 출력하는 역할을 맡는다. 각 줄의 * 개수를 보면, i 가 1 일 때, 즉 첫 번째 줄일 때의 *의 개수는 1 개이다. 그리고 i 가 2 일 때, 두 번째 줄일 때는 *의 개수가 2 개이다. 이렇듯, 줄의 위치 i 에 따라 *의 개수도 동일하게 증가한다(* 개수 = 줄의 위치). 따라서 두 번째 반복 문을 만드는 것도 간단하다. I 가 증가함에 따라 *개수를 늘리면 된다. 따라서 j 는 보기 쉽게 1 부터 i 까지 반복을 하면서 *을 출력한다. 이렇게 해야 i 가 증가함에 따라 출력되는 *의 개수는 늘어난다.

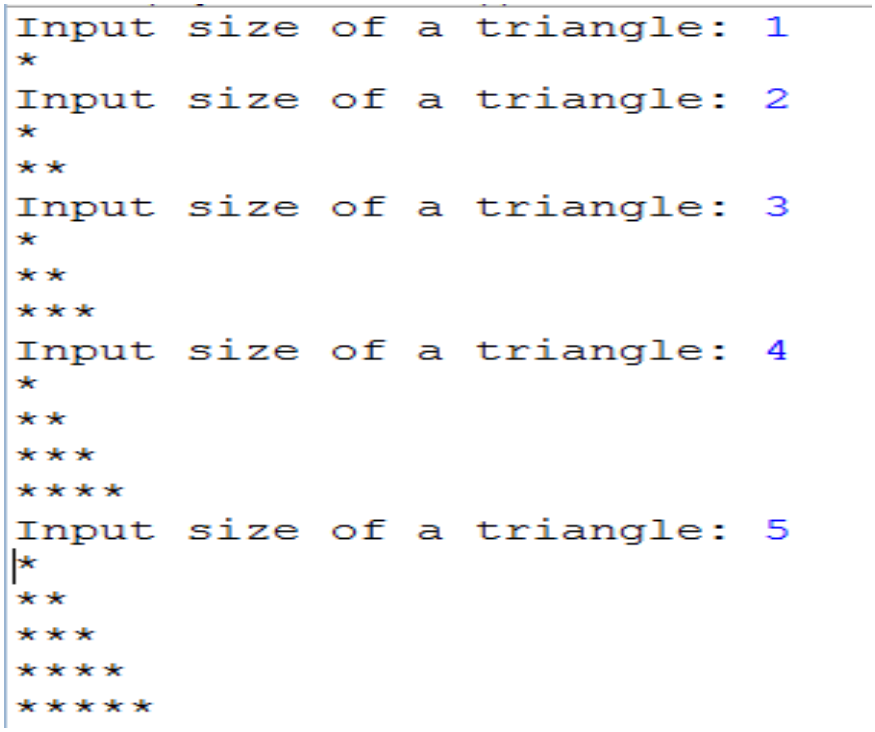
2.2 Source Code

```
void printTriangle()  //직각 삼각형 출력
{
    int n, i, j; //n은 삼각형의 크기, i,j는 반복문을 위한 변수
    printf("Input size of a triangle: ");
    fflush(stdout);
    scanf("%d", &n);

    for (i = 1; i <= n; i++)
```

```
        {
                for (j = 1; j <= i; j++)
                        printf("*");
                printf("\n");
        }
}
```

2.3 Result (Snap Shot)



3. 역직각삼각형 출력하기

3.1 Solution

이 문제도 2 번 문제와 비슷한 형식으로 생각하면 편하다. 일단 첫 번째 반복문이 삼각형의 크기 n 만큼 반복하는 것은 변함이 없다. 중요한 건 *이 출력되는 두 번째 반복 문인데, 출력에도 볼 수 있듯이 이번엔 좌상 구간을 직각으로 하는 삼각형이다. 그렇기에 밑 변이 반복 문의 맨 처음을 담당하게 된다. 즉 두 번째 반복문은 i 가 1 일 때 n 개 만큼의 *을 출력해야 하고, i 가 증가함에 따라, (n-i)개의 *을 출력하는 형태를 가진다. 따라서 두 번째 반복 문을 2 번과는 반대로 j 가 1 부터 (n-i)의 범위를 주게 되면, 자동으로 i 가 증가함에 따라 *의 개수 또한 줄어든다.

3.2 Source code

```
void printReverseTriangle() //역직각삼각형 출력
{
	int n, i, j; //n은 삼각형의 크기, i,j는 반복문을 위한 변수
	printf("Insert the size of a triangle: ");
	fflush(stdout);
	scanf("%d", &n);

	for (i = 1; i <= n; i++)
	{
		for (j = 0; j <= n - i; j++)
		{
			printf("*");
		}
		printf("\n");
	}
}
```

3.3 Result (Snap Shot)

```
Insert the size of a triangle: 2
**
*
Insert the size of a triangle: 4
****
***
**
*
Insert the size of a triangle: 5
*****
****
***
**
*
Insert the size of a triangle: 6
******
*****
****
***
**
*
```

```
Insert the size of a triangle: 10
**********
*********
********
*******
******
*****
****
***
**
*
```

4. 정삼각형 출력하기

4.1 Solution

이번 문제 또한 마찬가지로, 각 줄의 대한 *의 개수를 확인하면 접근하기 편하다. 2 와 3 번 문제와는 다르게 삼각형이지만, *이 1 칸씩 빈 채로 출력되는 것을 확인할 수 있다. 그리고 *의 개수 또한 줄이 증가함에 따라 1 개씩 늘어나는 것은 동일하다. 2 번과 3 번과 동일하게 출력했다면 *의 개수는 2 개씩 늘어났을 것이다. 여하튼 그것은 중요하지 않고, 또 하나 살펴볼 것이 시작점이 바로 n 크기의 삼각형 중앙에 있다는 것이다. 이를 해결하기 위해 첫 번째로는 두 번째 반복 문안을 for 문을 2 개를 사용해야 한다는 것이다. 그래서 첫 번째 for 문은 빈칸을 출력하고, 두 번째 for 문에는 *을 출력하게 하는 것이 좋다.

첫 번째 for 문을 구성하기 위해서는 일단 반복 문의 범위를 살펴보는 것이 좋다. 일단 처음부터 빈칸을 넣어야 하기 때문에 j = 1 에서 시작하는 것이 맞고, 범위를 살펴보면, i(줄)이 증가함에 따라 빈칸의 개수 또한 i 만큼 감소한다. 따라서 범위는 삼각형의 중앙인 n 에서 i 개를 뺀, (n-i)가 맞다. 이렇게 해야 줄이 증가함에 따라서 알맞은 개수의 띄어쓰기를 할 수 있다.

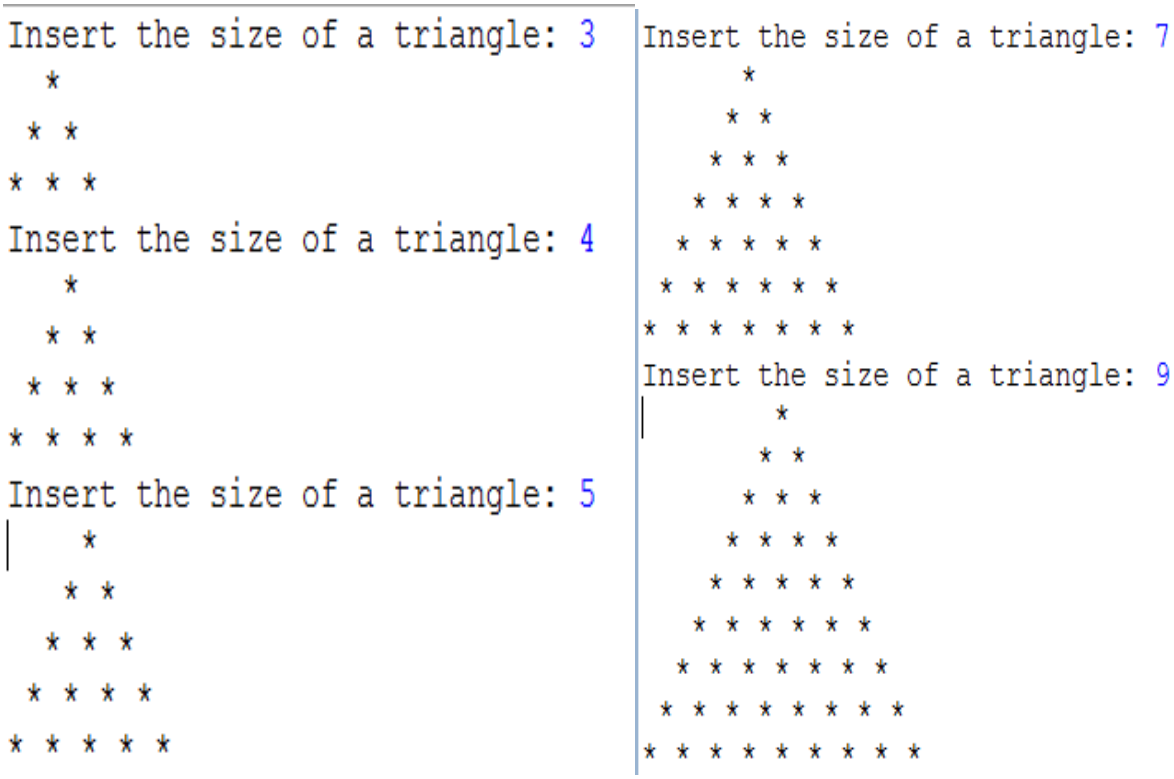
띄어쓰기 작업을 끝낸 후에는 *을 출력해야 한다. *의 개수를 일단 살펴보면, 전과 같이 i 가 증가함에 따라 *의 개수 또한 i 개 만큼 증가한다. 따라서 j 가 *의 개수만큼 반복할 수 있게 1 부터 i 까지 범위를 정해주면 된다. 여기서 중요한 것이 삼각형을 이루는 *이 한 칸씩 뛰고 출력되기 때문에, *을 출력할 때도 모양을 맞춰주기 위해, "* "식으로 출력을 해야한다. 그래야 출력 시에 모양이 알맞게 나온다.

4.2 Source Code

```
void printRealTriangle() //정삼각형 출력
{
	int n, i, j; //n은 삼각형의 크기, i,j는 반복을 위한 변수
	printf("Insert the size of a triangle: ");
	fflush(stdout);
	scanf("%d", &n);
```

```
        for (i = 1; i <= n; i++)
        {
                for (j = 1; j <= n - i; j++)
                        printf(" ");
                for (j = 1; j <= i; j++)
                        printf("* ");
                printf("\n");
        }
}
```

4.3 Result (Snap Shot)



5. 역 정삼각형 출력하기

5.1 Solution

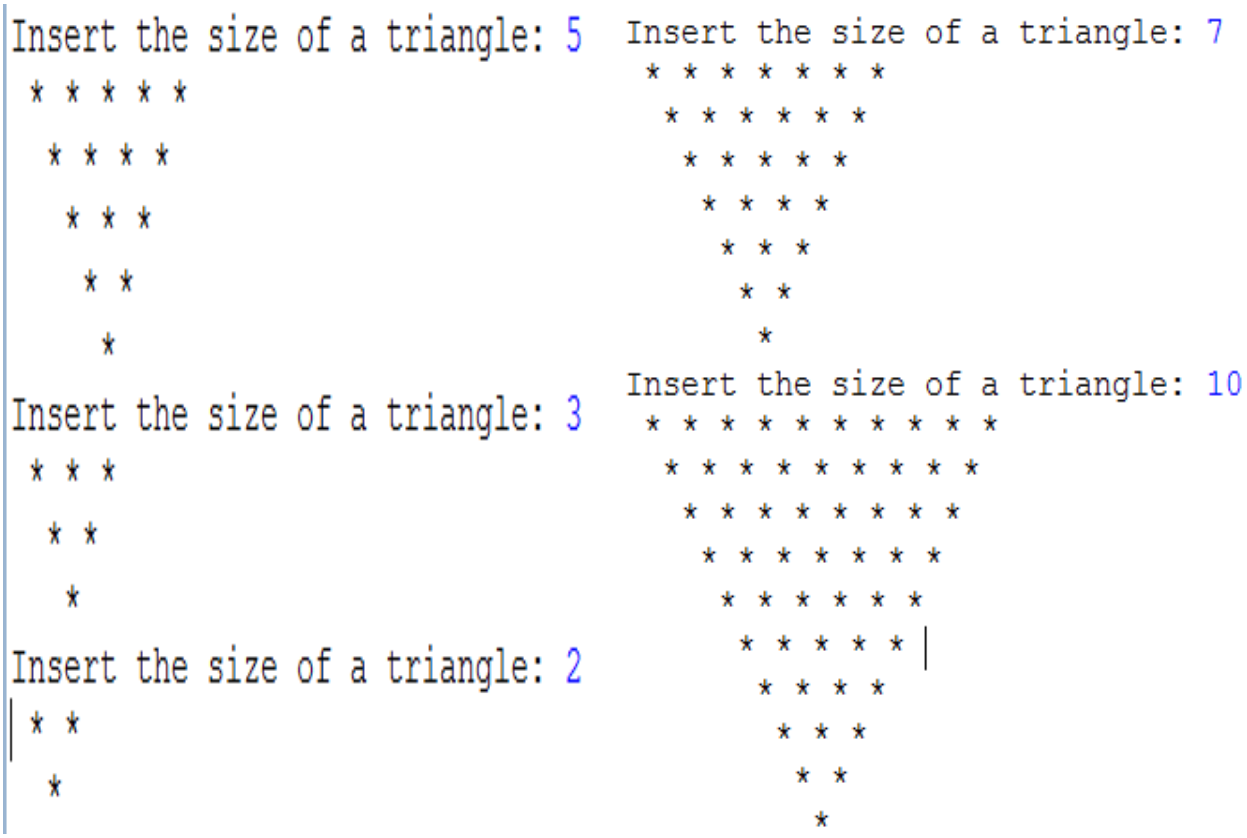
조금만 생각하면 답이 나오는 문제이다. 4 번에서는 별의 개수가 줄과 함께 늘어나지만, 이번 문제는 별의 개수가 줄어든다. 따라서 2 개의 for 문의 범위만 바꿔주면 간단하다. 빈 칸은 줄이 증가함에 따라 i 개 만큼 늘어나고, *은 i 개 만큼 줄어든다. 따라서 이에 알맞은 범위를 지정해 주면 마무리된다.

5.2 Source Code

```
void printRealReverseTriangle() //역정삼각형 출력
{
	int n, i, j; //n은 삼각형의 크기, i,j는 반복을 위한 변수
	printf("Insert the size of a triangle: ");
	fflush(stdout);
	scanf("%d", &n);

	for (i = 1; i <= n; i++)
	{
		for (j = 1; j <= i; j++)
			printf(" ");
		for (j = 0; j <= n - i; j++)
			printf("* ");
		printf("\n");
	}
}
```

5.3 Result (Snap Shot)



6. 마름모 출력하기

6.1 Solution

사실 딱히 설명할 것이 없다. 간단하게 4 번과 5 번을 합쳐주면 되는데, 주의할 것이 마름모의 중앙을 위쪽의 정삼각형이 출력할 것인지 아래쪽의 정삼각형이 출력할 것 인지만 정해주면 된다. 쉽게 말하면, 크기가 10 인 마름모는 크기가 5 인 정삼각형과 크기가 1 개 줄어든 4 의 역정삼각형을 합한 것과 같고, 다르게 생각하면 크기가 4 인 정삼각형과 5 인 역정삼각형과 같은 것이다. 따라서 위에서 5 를 출력하건 아래서 4 를 출력하건 중요하지 않지만, 그에 따라서 범위를 1 씩 줄여 주는 걸 잊으면 안된다. 그대로 n 의 크기로 각각 합치게 되면 마름모의 중앙 줄이 2 번 출력되는 사태가 일어난다.

6.2 Source Code

```
void printDiamond() //마름모 출력하기
{
	int n, i, j; //n은 삼각형의 크기, i,j는 반복을 위한 변수
	printf("Insert the size of a triangle: ");
	fflush(stdout);
	scanf("%d", &n);

	//마름모의 윗쪽 면(중간까지 포함)
	for (i = 1; i <= n; i++)
	{
		for (j = 1; j <= n - i; j++)
			printf(" ");
		for (j = 1; j <= i; j++)
			printf("* ");
		printf("\n");
	}

	//마름모의 아랫쪽 면(중간 포함 안됨)
	for (i = 1; i <= n - 1; i++)
	{
		for (j = 1; j <= i; j++)
			printf(" ");
```

```
            for (j = 0; j <= n - 1 - i; j++)
                printf("* ");

            printf("\n");
        }
}
```

6.3 Result (Snap Shot)

```
Insert the size of a triangle: 3
  *
 * *
* * *
 * *
  *
Insert the size of a triangle: 4
   *
  * *
 * * *
* * * *
 * * *
  * *
   *
Insert the size of a triangle: 5
    *
   * *
  * * *
 * * * *
* * * * *
 * * * *
  * * *
   * *
    *
```

```
Insert the size of a triangle: 9
        *
       * *
      * * *
     * * * *
    * * * * *
   * * * * * *
  * * * * * * *
 * * * * * * * *
* * * * * * * * *
 * * * * * * * *
  * * * * * * *
   * * * * * *
    * * * * *
     * * * *
      * * *
       * *
        *
```

```
Insert the size of a triangle: 7
          *
        * *
      * * *
    * * * *
  * * * * *
* * * * * * *
  * * * * *
    * * * *
      * * *
        * *
          *
```