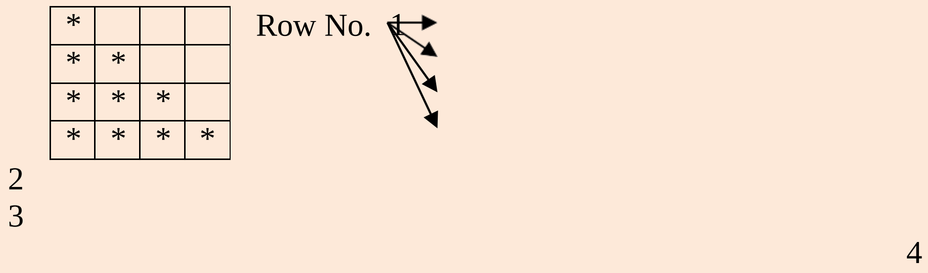
PATTERNS

A pattern is nothing but different types of designs that can be printed on the screen based on some user input. If somebody can print patterns like the ones shown in this chapter, he or she has the ability to master the concept of loops, especially the nested loop.

Say for example, we have to print a pattern like the following. If the user input a value, suppose 4, into a variable n, the program prints the following pattern:

```
*
**
***
****
```

To solve this problem, consider the pattern by plotting in into a graph paper. The trends of this pattern will be revealed right then.



This pattern now shows a few trends: the number of rows in the pattern is equal to the number entered by the user, and the number of stars in each row is equal to the row number.

As a result of this understanding, we can code the following program.

```
#include<stdio.h>

void main(void)
{
        int r, star, n;

        clrscr();

        printf("\n\tEnter a number : ");
        scanf("%d", &n);

        for(r = 1 ; r <= n ; r++)
        {
                printf("\n\t");
                for(star = 1 ; star <= r ; star++)
                        printf("*");
        }

        getch();
}
```

The next program is about a pyramid of stars.

			*			
		*	*	*		
	*	*	*	*	*	
*	*	*	*	*	*	*

1

2
3
4

The initial concept about the number of rows being equal to the number entered by the user remains the same. However, each loop contains some space as well as some stars. The number of space to be printed in each row is equal to the difference between the user entered number and the number of that row. The number of stars to be printed at each row is equal to one less than twice the number of that row.

```
#include<stdio.h>

void main(void)
{
	int r, star, space, n;

	clrscr();

	printf("\n\tEnter a number : ");
	scanf("%d", &n);

	for(r = 1 ; r <= n ; r++)
	{
		printf("\n\t");
		for(space = 1 ; space <= n-r ; space++)
			printf(" ");
		for(star = 1 ; star <=2*r-1 ; star++)
			printf("*");
	}

	getch();
}
```

In case where we are interested in printing a diamond of stars, the pattern would look like the following one.

			*			
		*	*	*		
	*	*	*	*	*	
*	*	*	*	*	*	*
	*	*	*	*	*	
		*	*	*		
			*			

1

2
3
4
3
2
1

The top half of the pattern (from row 1 to row 4) has already been printed by the previous program. The bottom half of the program, that is, the last three rows are nothing but the mirror image of the top three rows. The previous program can be modified a little bit to print a diamond.

```
#include<stdio.h>

void main(void)
{
        int r, star, space, n;

        clrscr();

        printf("\n\tEnter a number : ");
        scanf("%d", &n);

        for(r = 1 ; r <= n ; r++)
        {
                printf("\n\t");
                for(space = 1 ; space <= n-r ; space++)
                        printf(" ");
                for(star = 1 ; star <=2*r-1 ; star++)
                        printf("*");
        }

        for(r = n-1 ; r >= 1 ; r--)
        {
                printf("\n\t");
                for(space = 1 ; space <= n-r ; space++)
                        printf(" ");
                for(star = 1 ; star <=2*r-1 ; star++)
                        printf("*");
        }

        getch();
}
```

If we want to convert the diamond of stars into a diamond of integers, the pattern would look like:

			1			
		1	2	1		
	1	2	3	2	1	
1	2	3	4	3	2	1
	1	2	3	2	1	
		1	2	1		
			1			

1

2
3
4
3
2
1

```
#include<stdio.h>

void main(void)
{
        int r, star, space, n;

        clrscr();

        printf("\n\tEnter a number : ");
        scanf("%d", &n);

        for(r = 1 ; r <= n ; r++)
        {
                printf("\n\t");
                for(space = 1 ; space <= n-r ; space++)
                        printf(" ");
                for(star = 1 ; star <= r ; star++)
                        printf("%d", star);
                for(star = r-1 ; star >= 1 ; star--)
                        printf("%d", star);
        }

        for(r = n-1 ; r >= 1 ; r--)
        {
                printf("\n\t");
                for(space = 1 ; space <= n-r ; space++)
                        printf(" ");
                for(star = 1 ; star <= r ; star++)
                        printf("%d", star);
                for(star = r-1 ; star >= 1 ; star--)
                        printf("%d", star);
        }

        getch();
}
```

By modifying the above program just a little the diamond of numbers can be converted into a diamond of ASCII characters

			A			
		A	B	A		
	A	B	C	B	A	
A	B	C	D	C	B	A
	A	B	C	B	A	
		A	B	A		
			A			

1

2
3
4
3
2
1

```
#include<stdio.h>

void main(void)
{
        int r, star, space, n;

        clrscr();


        printf("\n\tEnter a number : ");
        scanf("%d", &n);

        for(r = 1 ; r <= n ; r++)
        {
                printf("\n\t");
                for(space = 1 ; space <= n-r ; space++)
                        printf(" ");
                for(star = 1 ; star <= r ; star++)
                        printf("%c", star+64);
                for(star = r-1 ; star >= 1 ; star--)
                        printf("%c", star+64);
        }

        for(r = n-1 ; r >= 1 ; r--)
        {
                printf("\n\t");
                for(space = 1 ; space <= n-r ; space++)
                        printf(" ");
                for(star = 1 ; star <= r ; star++)
                        printf("%c", star+64);
                for(star = r-1 ; star >= 1 ; star--)
                        printf("%c", star+64);
        }

        getch();
}
```

An interesting little program is the next one.

			1			
		2		3		
	4		5		6	
7		8		9		10

```
#include<stdio.h>

void main(void)
{
        int r, space, n, num = 1;

        clrscr();

        for(r = 1 ; r <= 4 ; r++)
        {
                printf("\n\t");
                for(space = 1 ; space <= 4-r ; space++)
                        printf(" ");
                for(n = 1 ; n <= r ; n++)
                        printf("%d ", num++);
        }

        getch();
}
```

In the next program we are creating a hollow diamond, surrounded by stars.

*	*	*	*	*	*	*
*	*	*		*	*	*
*	*				*	*
*						*
*	*				*	*
*	*	*		*	*	*
*	*	*	*	*	*	*

4

3
2
1
2
3
4

```
#include<stdio.h>

void main(void)
{
            int r, n, star, space;

            clrscr();

            printf("\n\tEnter a number : ");
            scanf("%d", &n);

            for(r = n ; r >= 1 ; r--)
            {
                        printf("\n\t");

                        if(r == n)
                        {
                                    for(star = 1 ; star <= 2*r-1 ; star++)
                                                printf("*");
                        }
                        else
                        {
                                    for(star = 1 ; star <= r ; star++)
                                                printf("*");
                                    for(space = 1 ; space <= 2*(n-r)-1 ; space++)
                                                printf(" ");
                                    for(star = 1 ; star <= r ; star++)
                                                printf("*");
                        }
            }

            for(r = 2 ; r <= n ; r++)
            {
                        printf("\n\t");

                        if(r == n)
                        {
                                    for(star = 1 ; star <= 2*r-1 ; star++)
                                                printf("*");
                        }
                        else
                        {
                                    for(star = 1 ; star <= r ; star++)
                                                printf("*");
                                    for(space = 1 ; space <= 2*(n-r)-1 ; space++)
                                                printf(" ");
                                    for(star = 1 ; star <= r ; star++)
                                                printf("*");
                        }
            }

            getch();
}
```

The final program prints the same hollow diamond, surrounded by ASCII characters. As you can guess, it could have been surrounded by integers too.

A	B	C	D	C	B	A
A	B	C		C	B	A
A	B				B	A
A						A
A	B				B	A
A	B	C		C	B	A
A	B	C	D	C	B	A

4

3
2
1
2
3
4

```
#include<stdio.h>

void main(void)
{
        int r, n, star, space;

        clrscr();

        printf("\n\tEnter a number : ");
        scanf("%d", &n);

        for(r = n ; r >= 1 ; r--)
        {
                printf("\n\t");

                if(r == n)
                {
                        for(star = 1 ; star <= r ; star++)
                                printf("%c", star+64);
                        for(star = r-1 ; star >= 1 ; star--)
                                printf("%c", star+64);
                }
                else
                {
                        for(star = 1 ; star <= r ; star++)
                                printf("%c", star+64);
                        for(space = 1 ; space <= 2*(n-r)-1 ; space++)
                                printf(" ");
                        for(star = r ; star >= 1 ; star--)
                                printf("%c", star+64);
                }
        }

        for(r = 2 ; r <= n ; r++)
        {
                printf("\n\t");

                if(r == n)
                {
                        for(star = 1 ; star <= r ; star++)
                                printf("%c", star+64);
                        for(star = r-1 ; star >= 1 ; star--)
                                printf("%c", star+64);
                }
                else
                {
                        for(star = 1 ; star <= r ; star++)
                                printf("%c", star+64);
                        for(space = 1 ; space <= 2*(n-r)-1 ; space++)
                                printf(" ");
                        for(star = r ; star >= 1 ; star--)
```

```
                                printf("%c", star+64);
                    }
            }

            getch();
    }
```