Write a program to find the value of one number raised to the power of another.

- $2^5 = 2*2*2*2*2$
- $5^7 = 5*5*5*5*5*5*5$

Logic

Initializing the count

Checking the condition

Increment/ Decrement

Write a program to find the value of one number raised to the power of another.

- $2^{5 =}$ 2*2*2*2*2

- 5^{7} = 5*5*5*5*5*5*5

Logic

Initializing the count	count = 1
Checking the condition	while(count<=pow)
Increment/ Decrement	count++

Write a program to find the value of one number raised to the power of another.

- 2^{5} = 2*2*2*2*2
- 5^{7} = 5*5*5*5*5*5*5

Logic

Initializing the count count = 1

Checking the condition while(count<=pow)

Increment/ Decrement count++

Body of the loop

Product=1		
Product = 1 * 2	= 2	count=1
Product = 2*2	=4	count=2
Product = 4*2	=8	count=3
Product = 8*2	=16	count=4
Product = 16*2	=32	count=5
Product = Product *num		

Program 3# Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.

Program 3# Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.

Algorithm

Step 1: Initialize positive =0 , negative=0 , zero=0

Step 2: Initialize choice = ‘y’

Step 3: while choice ==‘ y’

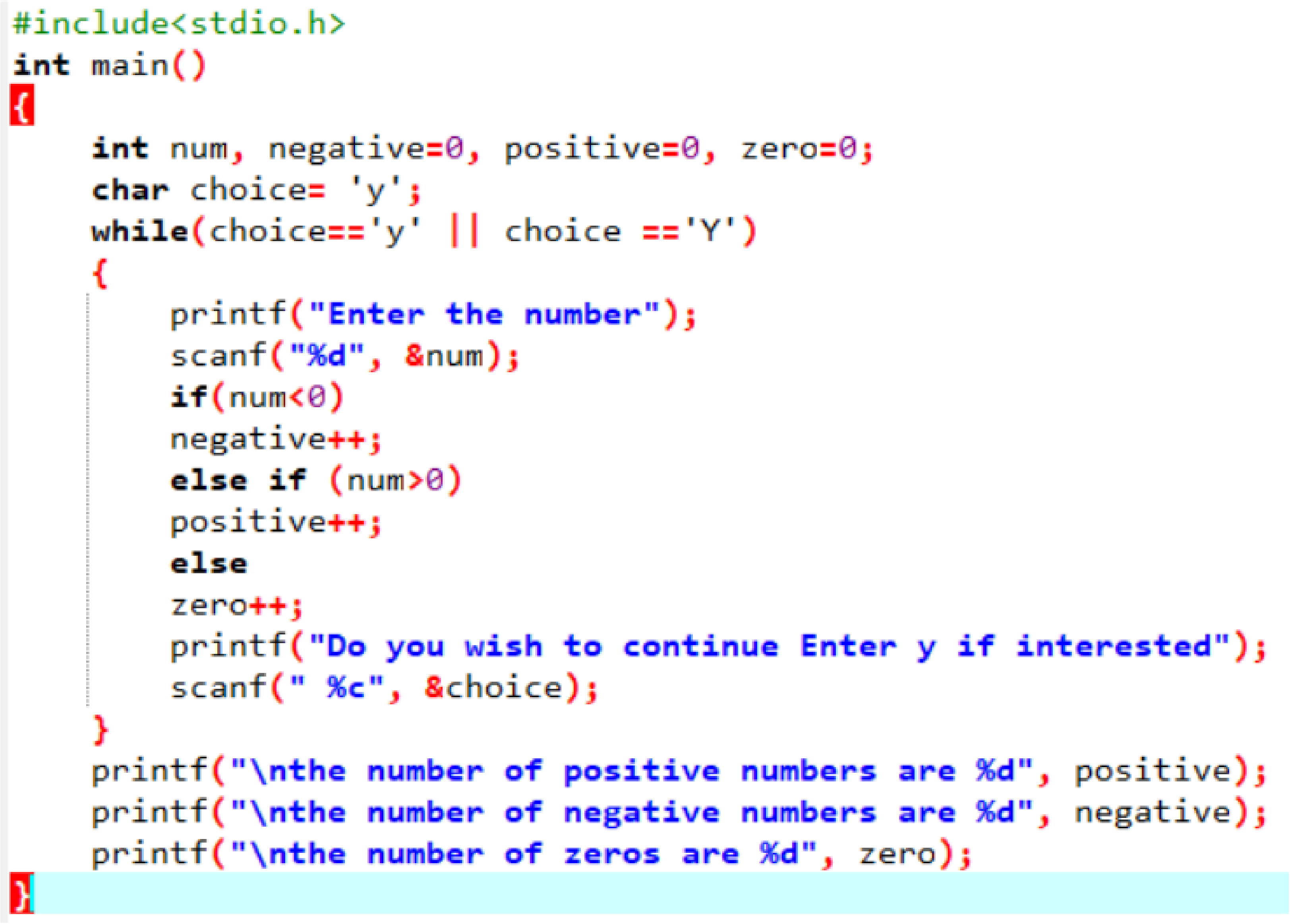
Step 4: Enter the number

- If number < 0 then increment negative by 1
- else if number = 0 then increment zero by 1
- else if number > 0 then increment positive by 1
- Step 5: Do you wish to enter another number – if yes enter choice as ‘y’ and go to step 3
- Step 6: End

Program 3# Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.

Algorithm

```
Step 1: Initialize positive =0 , negative=0 , zero=0
Step 2: Initialize choice = ‘y’    // Initialize the counter
Step 3: while choice ==‘ y’        // Condition checking
Step 4: Enter the number
•   If number < 0 then increment negative by 1
•   else if number = 0 then increment zero by 1
•   else if number > 0 then increment positive by 1
•   Step 5: Do you wish to enter another number – if yes enter
    choice as ‘y’ and go to step 3              // Changing the
    counter
•   Step 6: End
```



```
#include<stdio.h>
int main()
{
    int num, negative=0, positive=0, zero=0;
    char choice= 'y';
    while(choice=='y' || choice =='Y')
    {
        printf("Enter the number");
        scanf("%d", &num);
        if(num<0)
        negative++;
        else if (num>0)
        positive++;
        else
        zero++;
        printf("Do you wish to continue Enter y if interested");
        scanf(" %c", &choice);
    }
    printf("\nthe number of positive numbers are %d", positive);
    printf("\nthe number of negative numbers are %d", negative);
    printf("\nthe number of zeros are %d", zero);
}
```

Program #4 . Write a program to find sum of the digits of a number

Algorithm:-for a five digit number

```
Step1: Enter a 5 digit number ‘n’.
Step 2: Initialize sum =0.
Step 3: b=n%10. n=n/10; sum =sum+b.
Step 4: b=n%10. n=n/10; sum =sum+b.
Step 5: b=n%10. n=n/10; sum =sum+b.
Step 6: b=n%10. n=n/10; sum =sum+b.
Step 7: b=n%10. n=n/10; sum =sum+b.
Step 8: Print sum.
```

Initialize the counter
Condition checking
Loop counter
Body of the loop

```
    b=n%10
    n=n/10
    sum=sum+b
```

Algorithm:-for a five digit number

```
Step1: Enter a 5 digit number ‘n’.
Step 2: Initialize sum =0.
Step 3: b=n%10. n=n/10; sum =sum+b.
Step 4: b=n%10. n=n/10; sum =sum+b.
Step 5: b=n%10. n=n/10; sum =sum+b.
Step 6: b=n%10. n=n/10; sum =sum+b.
Step 7: b=n%10. n=n/10; sum =sum+b.
Step 8: Print sum.
```

Initialize the counter

```
        count=1
```

Condition checking

```
        while(count<=5)
```

Loop counter

```
        count++
```

Body of the loop

```
        b=n%10
        n=n/10
        sum=sum+b
```

Algorithm:-for a n digit number

```
Step1: Enter a number ‘n’.
Step 2: Initialize sum =0.
Step 3: Repeat the steps 4 to 6 while (n) is true
Step 4: b=n%10.
Step 5: n=n/10;
Step 6: sum =sum+b and go to step 3
Step 7: Print sum.
```

Initialize the counter

n   (*n is the number*)

Condition checking

while(n)

Loop counter

Body of the loop

```
b=n%10
n=n/10
sum=sum+b
```

Program 1# Write a program to find whether the given number is Armstrong or not

Armstrong Number

- **Armstrong number** is a **number** that is equal to the sum of cubes of its digits. For example 0, 1, 153, 370, 371 and 407, 1634, 8208, 9474, 54748, ... are the **Armstrong numbers**.
- Ex: $1^3 + 5^3 + 3^3 =153$

Number is Armstrong or not

```
Step 1: Enter the number num
Step 2: Initialize sum=0, num1=num
Step 3: Use while loop to extract individual digits and repeat steps 4 to 6
Step 4: a= num%10
Step 5: sum + = a*a*a
Step 6: num=num/10
Step 7: if(sum is equal to num1)
            Print it is an Armstrong number
        else
            it is not an Armstrong number
```

Number is Armstrong or not

```
Step 1: Enter the number num
Step 2: Initialize sum=0, num1=num
Step 3: Use while loop to extract individual
 digits and repeat steps 4 to 6
Step 4: a= num%10
Step 5: sum + = a*a*a
Step 6: num=num/10
Step 7: if(sum is equal to num1)
          Print it is an Armstrong number
          else
          it is not an Armstrong number
```

153

Sum=0, num1=153

while(num)

Iteration 1:

a=153%10 = 3

sum=27

num=15

while(15)

Iteration 2:

a=15%10 = 5

sum=27 + 125 = 152

num=1

while(1)

Iteration 3:

a=1%10 = 1

sum=152 + 1 = 153

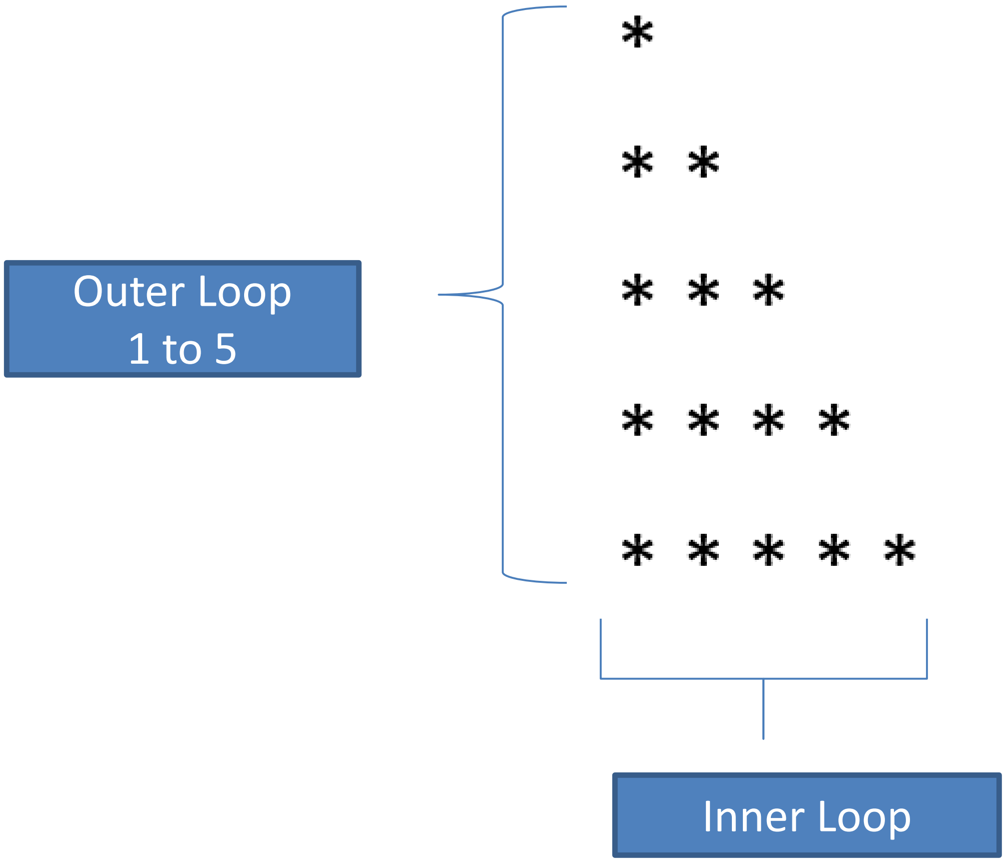
num=0

while(0)

Program 2# Write a program to generate the series

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

Series //Nested for loop

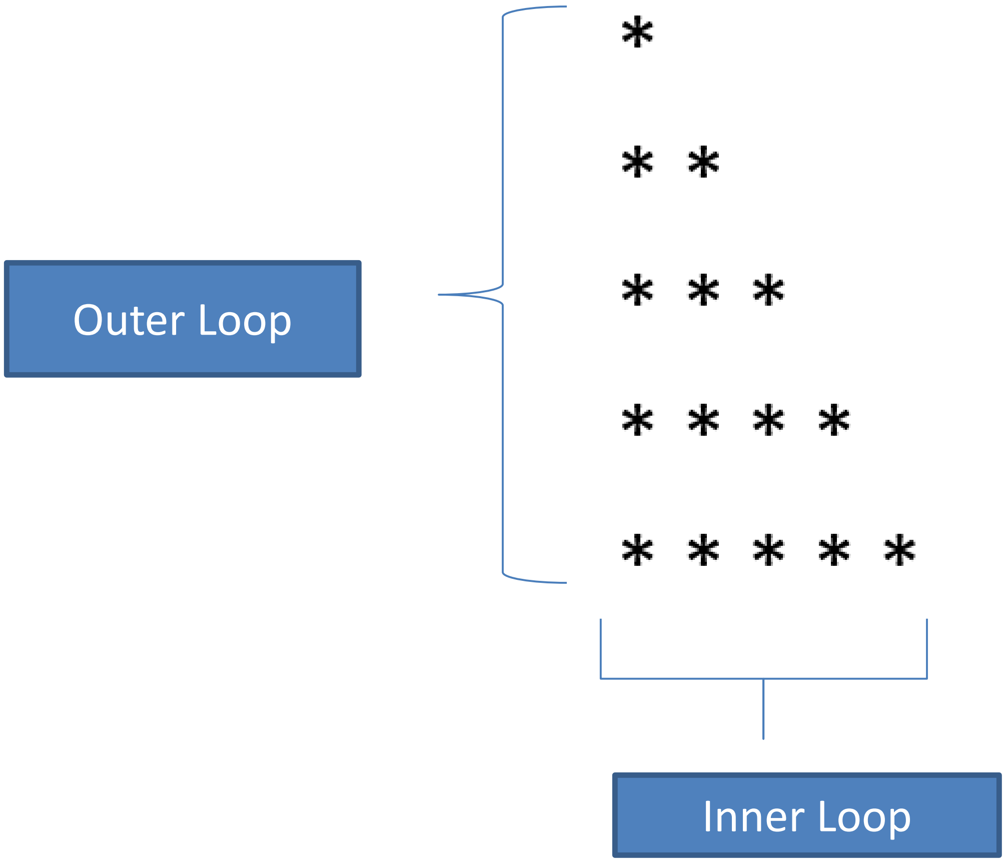


```
*
* *
* * *
* * * *
* * * * *
```

Outer Loop
1 to 5

Inner Loop

Series

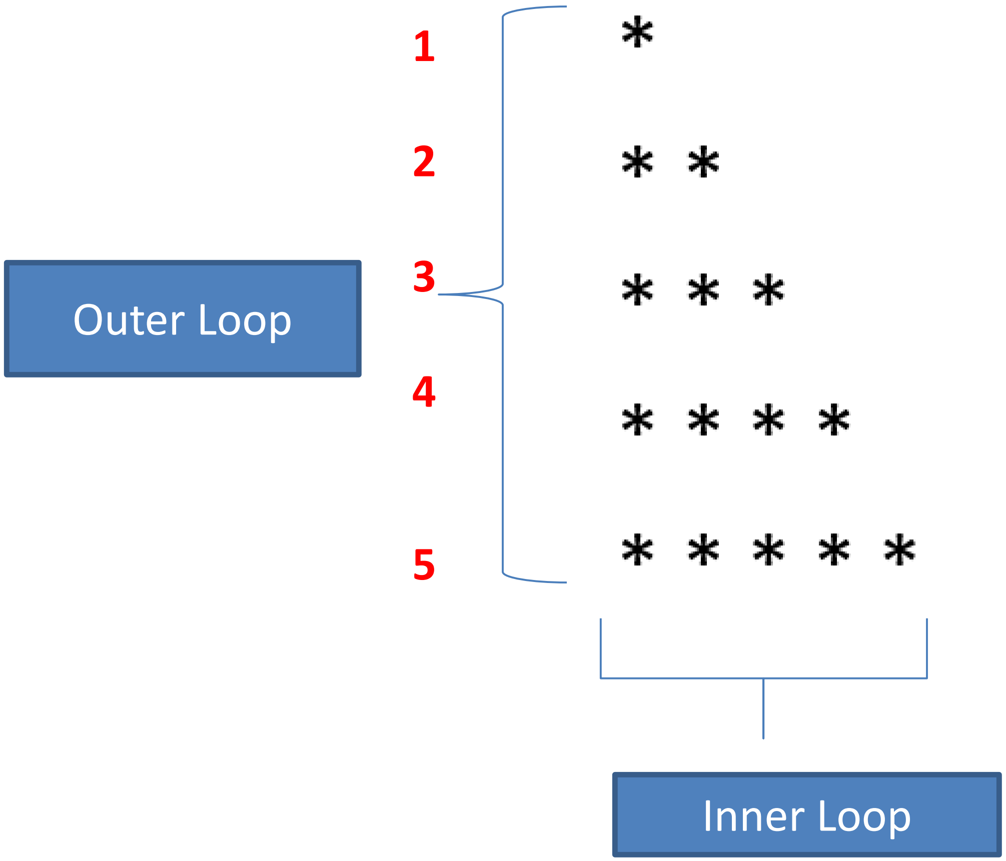


Establish Relationship

Outer Loop – Inner Loop

Step 1	*
Step 2	**
Step 3	***
Step 4	****
Step 5	*****

Series



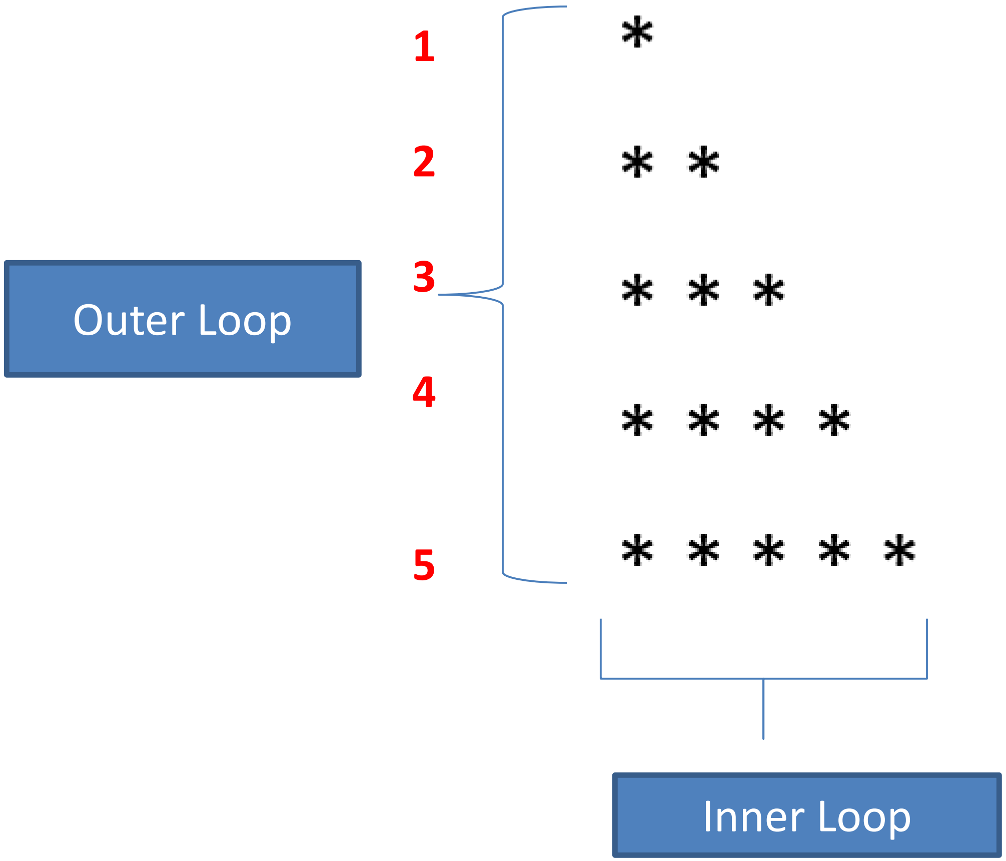
Establish Relationship

Outer Loop – Inner Loop

Step 1	*
Step 2	**
Step 3	***
Step 4	****
Step 5	*****

```
for(k=1;k<=5;k++)
{
 for(j=1;j<=k;j++)
 {
   printf(“*”);
  }
}
```

Series



Establish Relationship

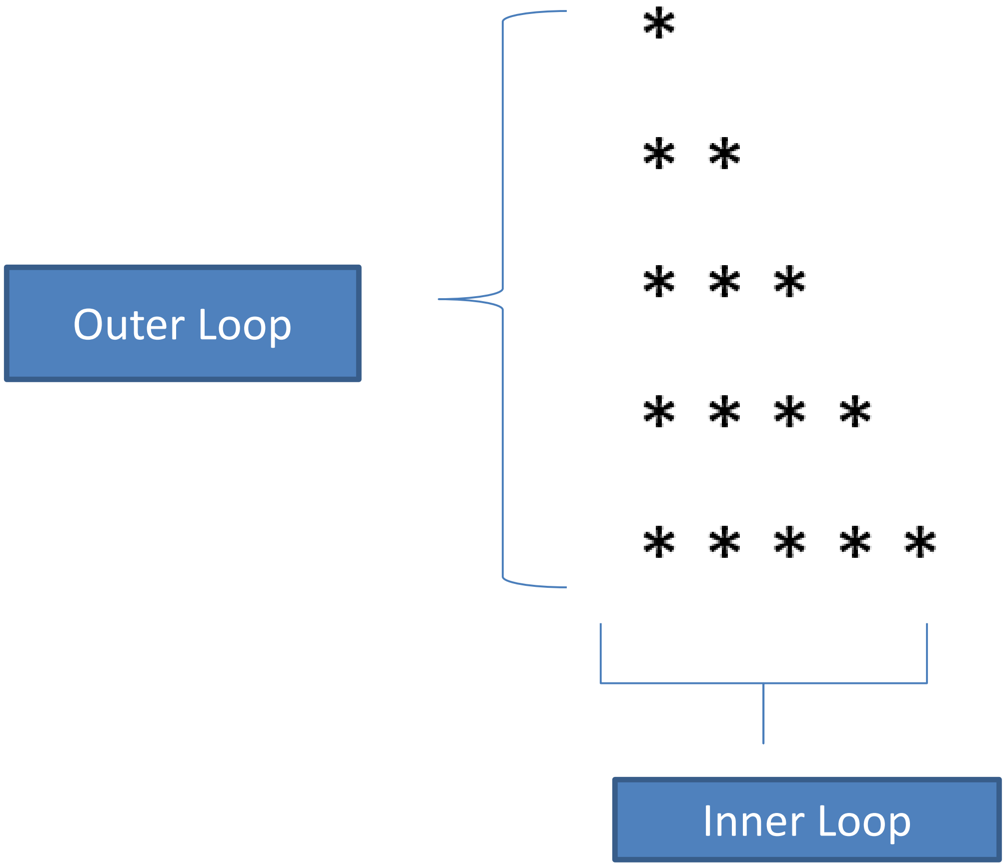
Outer Loop – Inner Loop

Step 1	*
Step 2	**
Step 3	***
Step 4	****
Step 5	*****

```
for(k=1;k<=5;k++)
{
 for(j=1;j<=k;j++)
 {
   printf("*");
  }
}
```

Output: ***************

Series



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

Outer Loop

Inner Loop

Establish Relationship

Outer Loop – Inner Loop

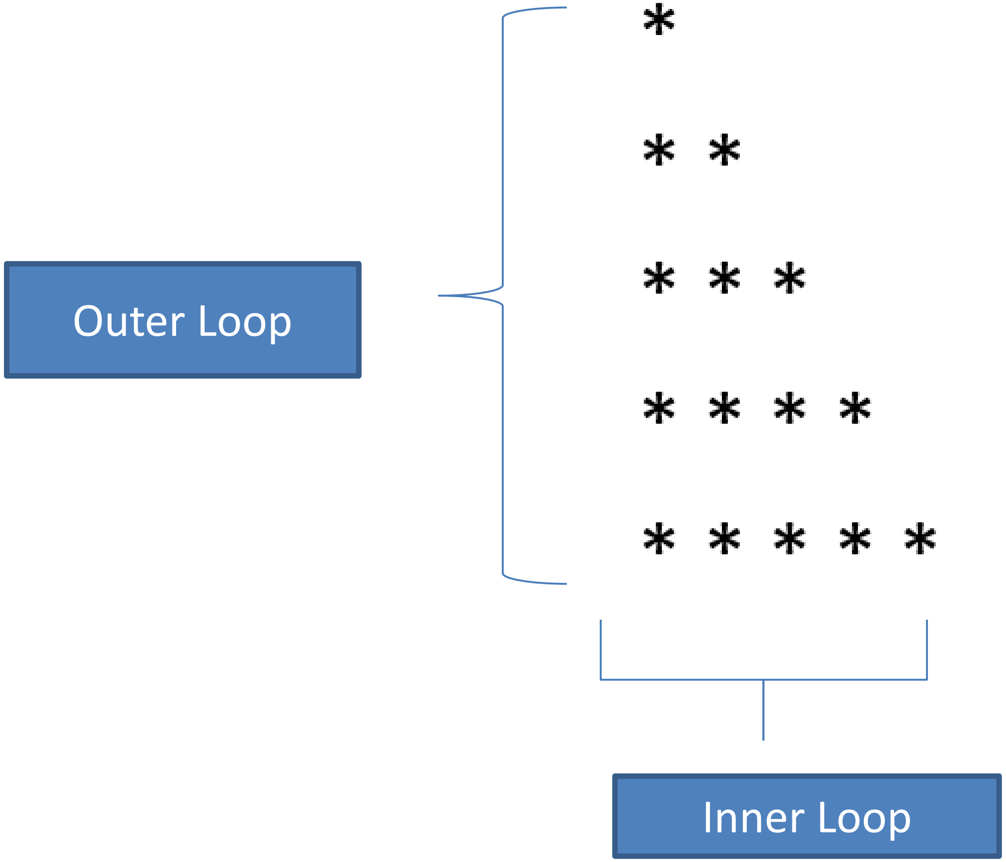
Step 1	*
Step 2	**
Step 3	***
Step 4	****
Step 5	*****

```
for(k=1;k<=5;k++)
{
  for(j=1;j<=k;j++)
  {
     printf(“*\n”);
  }
}
```

Output:

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

Series



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

Outer Loop

Inner Loop

Establish Relationship

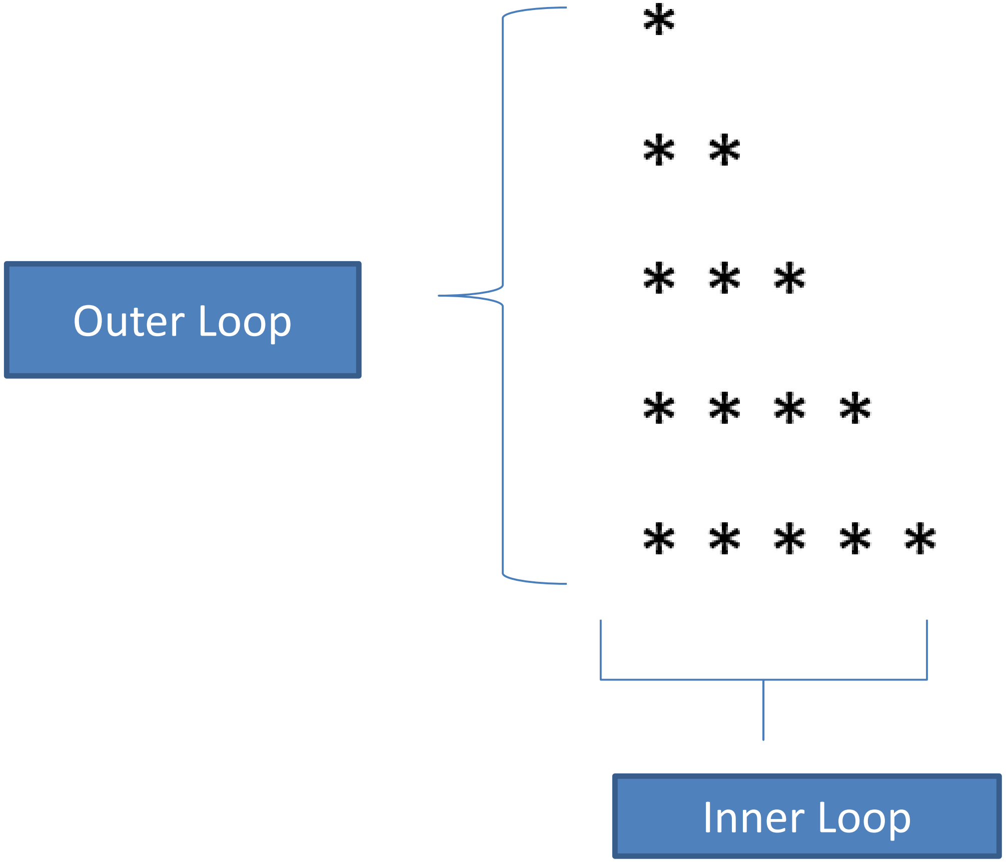
Outer Loop –	Inner Loop
Step 1	*
Step 2	**
Step 3	***
Step 4	****
Step 5	*****

```
for(k=1;k<=5;k++)
{
 for(j=1;j<=k;j++)
 {
   printf(“*”);
  }
  printf(“\n”);
}
```

Output:

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

Series



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

Establish Relationship

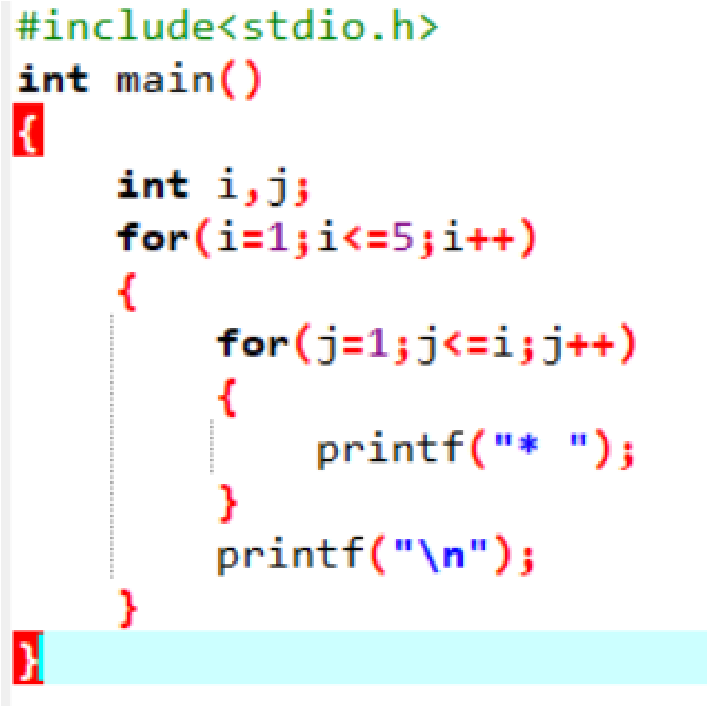
Outer Loop – Inner Loop	
Step 1	*
Step 2	**
Step 3	***
Step 4	****
Step 5	*****

```
for(k=1;k<=5;k++)
{
 for(j=1;j<=k;j++)
 {
   printf("* ");
  }
  printf("\n");
}
```

Output:

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

Series generation



```
#include<stdio.h>
int main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=1;j<=i;j++)
        {
            printf("* ");
        }
        printf("\n");
    }
}
```

Program 3# Write a program to generate the series

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

Establish Relationship

Outer Loop – Inner Loop

Step 1	1
Step 2	12
Step 3	123
Step 4	1234
Step 5	12345

```
for(k=1;k<=5;k++)
{
 for(j=1;j<=k;j++)
 {
   printf(“j ”);
  }
  printf(“\n”);
}
```

Program 3# Write a program to generate the series

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Establish Relationship

Outer Loop – Inner Loop

Step 1	1
Step 2	12
Step 3	123
Step 4	1234
Step 5	12345

```
for(k=1;k<=5;k++)
{
  for(j=1;j<=k;j++)
  {
    printf(“j ”);
  }
  printf(“\n”);
}
```

```
J
J J
J J J
J J J J
J J J J J
```

Program 3# Write a program to generate the series

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Establish Relationship

Outer Loop – Inner Loop	
Step 1	1
Step 2	12
Step 3	123
Step 4	1234
Step 5	12345

```
for(k=1;k<=5;k++)
{
  for(j=1;j<=k;j++)
  {
     printf("%d ",j);
   }
   printf("\n");
}
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Program 3# Write a program to generate the series

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Establish Relationship

Outer Loop – Inner Loop

Step 1	1
Step 2	12
Step 3	123
Step 4	1234
Step 5	12345

```
for(k=1;k<=5;k++)
{
  for(j=1;j<=k;j++)
  {
     printf("%d ",j);
   }
   printf("\n");
}
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Program 4# Write a program to generate the series

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Establish Relationship

Outer Loop – Inner Loop

Step 1	1
Step 2	12
Step 3	123
Step 4	1234
Step 5	12345

```
for(k=1;k<=5;k++)
{
  for(j=1;j<=k;j++)
  {
    printf(“%d ”,j);
  }
  printf(“\n”);
}
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Establish Relationship

Outer Loop – Inner Loop

Step 1	1
Step 2	22
Step 3	333
Step 4	4444
Step 5	55555

```
for(k=1;k<=5;k++)
{
  for(j=1;j<=k;j++)
  {
    printf(“%d ”,k);
  }
  printf(“\n”);
}
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Program #5 Write a program to generate the series

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

Outer for loop 1
Inner for loop 2
first inner loop prints spaces
Second inner loop prints *

```
for(k=1;k<=5;k++)
{
First inner loop for printing spaces
  for(j=1;j<=k;j++)
  {
    printf("* ");
   }
   printf("\n");
}
```

Outer loop

Inner loop 1

Inner loop 2

Program #5 Write a program to generate the series

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

Outer for loop 1

Inner for loop 2

first inner loop prints spaces

Second inner loop prints *

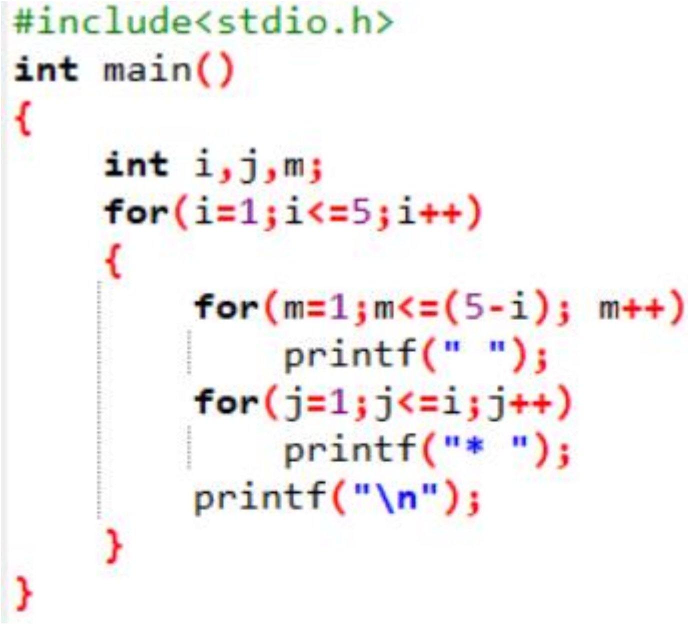
```
for(k=1;k<=5;k++)
{
  for(m=1;m<=(5-k); m++)
    {
      printf(“ “);
    }
  for(j=1;j<=k;j++)
    {
      printf(“* ”);
     }
      printf(“\n”);
}
```

First inner loop for printing spaces

Establish relation

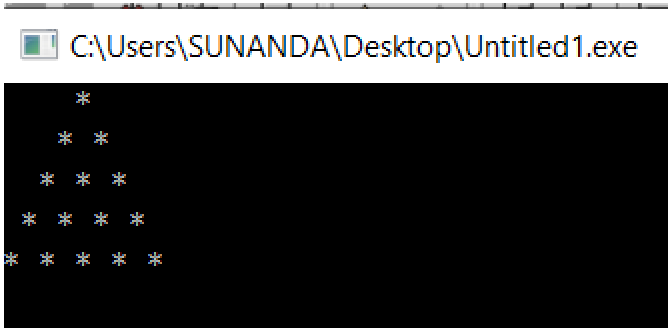
Outer- Inner loop

1 st loop	prints 4 spaces
2nd loop	prints 3 spaces
3rd loop	prints 2 spaces
4th loop	prints 1 space
5th loop	prints 0 space
ith loop prints	(5-i) spaces



```
#include<stdio.h>
int main()
{
	int i,j,m;
	for(i=1;i<=5;i++)
	{
		for(m=1;m<=(5-i); m++)
			printf(" ");
		for(j=1;j<=i;j++)
			printf("* ");
		printf("\n");
	}
}
```

```
for(k=1;k<=5;k++)
{
 for(m=1;m<=(5-i); m++)
  {
    printf(“ “);
  }
 for(j=1;j<=k;j++)
  {
    printf(“*”);
   }
    printf(“\n”);
}
```



C:\Users\SUNANDA\Desktop\Untitled1.exe

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

Program 6 #Program to print all Armstrong numbers between 1 and 1000

Single number is Armstrong or not

```
Step 1: Enter the number num
Step 2: Initialize sum=0, num1=num
Step 3: Use while loop to extract individual digits and repeat steps 4 to 6
Step 4: a= num%10
Step 5: sum + = a*a*a
Step 6: num=num/10
Step 7: if(sum is equal to num1)
            Print it is an Armstrong number
        else
            it is not an Armstrong number
```

Print all armstrong numbers between 1 and 1000

```
Step 1: Repeat steps 2 to 8 for (i=1;i<1000;i++)
Step 2: num =i
Step 3: Initialize sum=0, num1=num
Step 4: Use while loop to extract individual digits and repeat steps 5 to 7
Step 5: a= num%10
Step 6: sum + = a*a*a
Step 7: num=num/10
Step 8: if(sum is equal to num1)
            Print it is an Armstrong number
        else
            it is not an Armstrong number
```