Input and Output Function

- C programming has several in-built library functions to perform input and output tasks.
- The input/output functions are classified into two types
  - Formatted I/O functions
  - Unformatted I/O functions

Formatted I/O functions

- Two commonly used functions for formatted I/O (Input/Output) are printf() and scanf().
- The scanf() function reads formatted input from standard input (keyboard) whereas the printf() function sends formatted output to the standard output (screen).

printf() function

- The **printf() function** is used for output. It prints the given statement to the console.
- The syntax of printf() function is given below:
  - printf("format string",argument_list);
- The **format string** can be %d (integer), %c (character), %s (string), %f (float) etc.

Printf modifiers

%[flag][width][precision][size]conversion character

- **Width modifier**
  - specifies the width of the output
  - Example

```
    int a=10;
    printf("\n%d",a);//
printf("\n%d",12);
printf("\n%10d",12);
```

Example 1

```
#include<stdio.h>
 void main()
 {
   int num1 = 10;
   float num2 = 9.23;

   printf("num1 = %d\n",num1);
   printf("num1 = %5d\n",num1);
   printf("num1 = %10d\n",num1);

   printf("num2 = %f\n",num2);
   printf("num2 = %12f\n",num2);
}
```

num1	1	0

num1				1	0

num1									1	0

num2	9	.	2	3	0	0	0	0

num2					9	.	2	3	0	0	0	0

- Precision
  - Specifies the width after the decimal point
  - Example

```
      printf("\n%.3f",b);
 printf("\n%7.2f",b);
```

- Size modifier
  - h for short int
  - l for long int
  - ll for long int
  - L for long double

Example 2

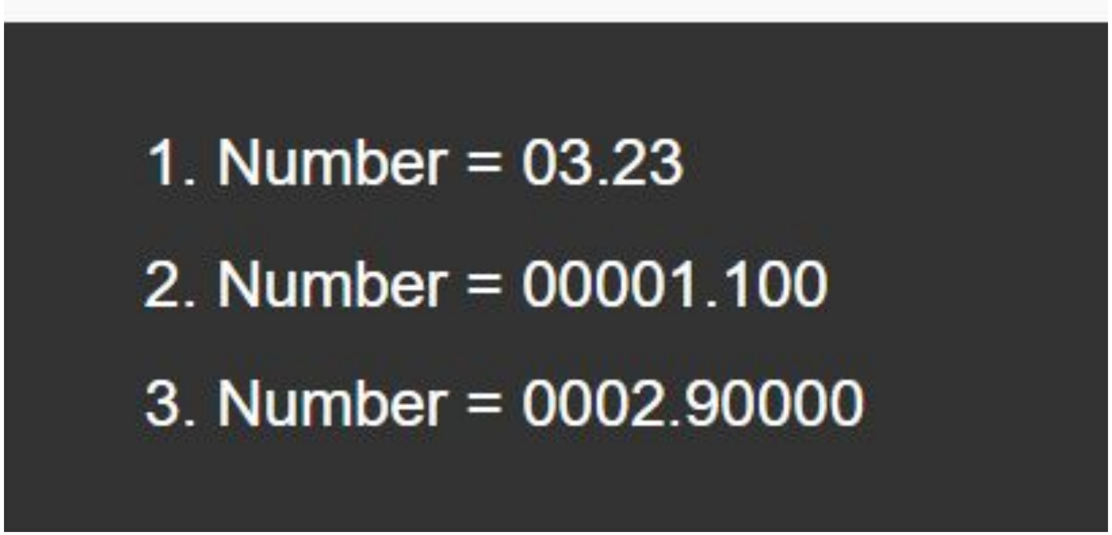
```
#include<stdio.h>
int main()
{
    short int n1 = 10;
    long int n2 = 100000;
    long long int n3 = 5060626500325;
    long double n4 = 556626.36595;
    printf("n1 = %hd\n",n1);
    printf("n2 = %ld\n",n2);
    printf("n3 = %lld\n",n3);
    printf("n4 = %Lf\n",n4);
    return 0;
}
```

- Flag modifier
- The flag modifier allows one or more print modifications to be specified. The flag can be any one of the characters from the below.
  - (- minus): left justify
  - (+ plus):prepends a plus for positive values
  - Space: prepends space for positive values
  - (0 zero): with width, prepends zero for numeric types
  - #: different advantages (hexadecimal,octal)
  - Example

```
printf("\n%010d",12);
```

Example 3
Pad with leading zeros

```
#include<stdio.h>
 int main()
 {
   printf("1. Number = %05.2f\n",3.23);
   printf("2. Number = %09.3f\n",1.1);
   printf("3. Number = %010.5f",2.9);
   return 0;
 }
```

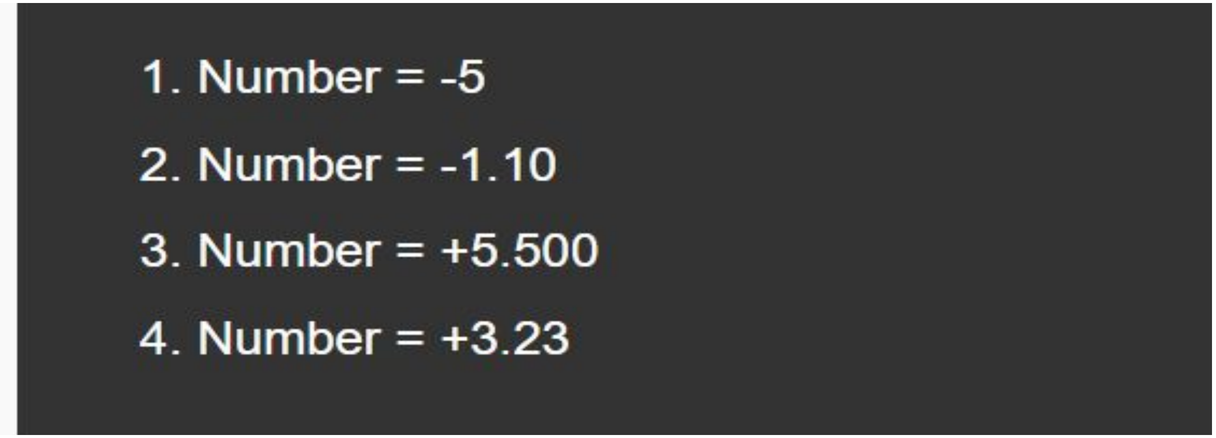


```
1. Number = 03.23
2. Number = 00001.100
3. Number = 0002.90000
```

Example 4

Display sign of the value

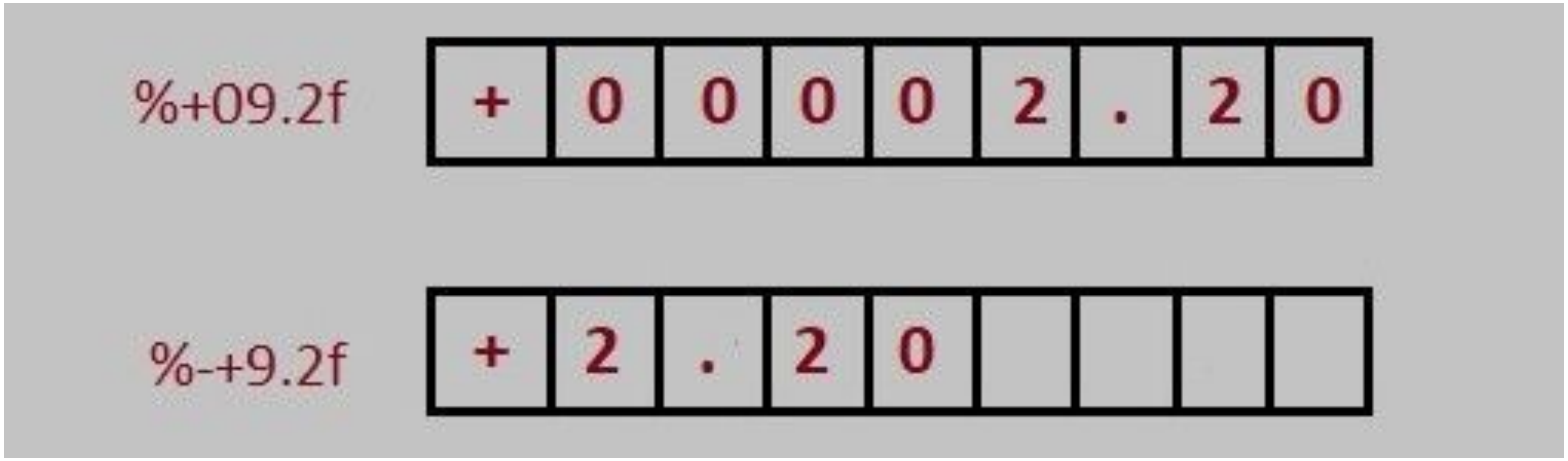
```
#include<stdio.h>
int main()
{
  printf("1. Number = %d\n",-5);
  printf("2. Number = %+.2f\n",-1.1);
  printf("3. Number = %+.3f\n",5.5);
  printf("4. Number = %+5.2f\n",3.23);
  return 0;
}
```



```
1. Number = -5

2. Number = -1.10

3. Number = +5.500

4. Number = +3.23
```

Example 5

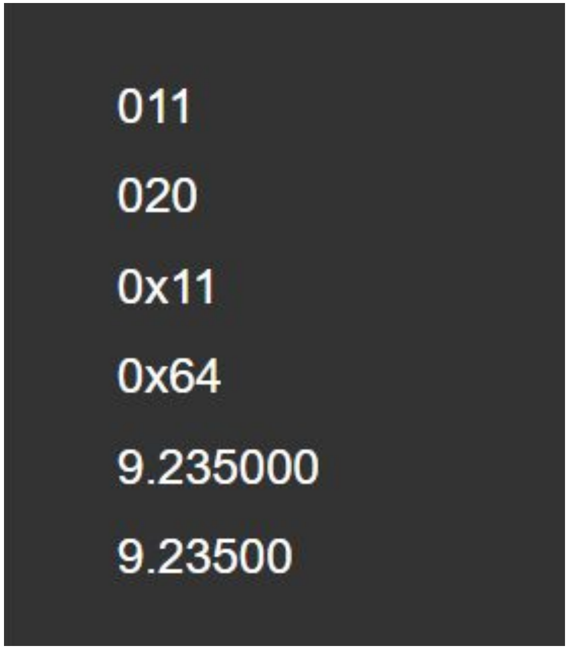
```
#include<stdio.h>
 void main()
 {
   // without flag 0
   printf("Number = %+09.2f\n",2.2);
   // with flag -
   printf("Number = %-+9.2f\n",2.2);
}
```



%+09.2f	+	0	0	0	0	2	.	2	0
%-+9.2f	+	2	.	2	0				

Example 6

```
#include<stdio.h>
 int main()
 {
   printf("%#o \n",9);
   printf("%#o \n",16);
   printf("%#x \n",17);
   printf("%#x \n",100);
   printf("%#f \n",9.235);
   printf("%#g \n",9.235);
   return 0;
 }
```



```
011
020
0x11
0x64
9.235000
9.23500
```

Example 7

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a=19;
    float b=18.5;
    printf("\n%d",a);
    printf("\n%d",12);
    printf("\n%10d",12);
    printf("\n%010d",12);

    int m=1256;
    printf("\n%d",m);
```

Example 8

```
#include<stdio.h>
void main()
{
      int a;
      a=printf("hello");
      printf("\nvalue return from printf is %d",a);//5

      a=printf("\nhello world");
      printf("\nvalue return from printf is %d",a);//12
}
```

scanf() function

- The **scanf() function** is used for input. It reads the input data from the console.
- scanf("format string",argument_list);

Unformatted I/O functions

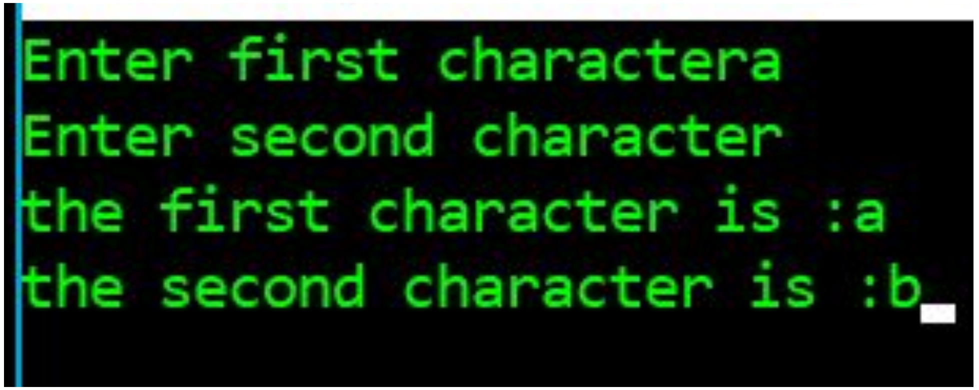
- Unformatted functions do not allow the user to read and display data in desired format.
- These library functions basically deal with a single character or a string of characters.
- The functions are
  - getchar()
  - putchar()
  - gets()
  - puts()
  - getch()
  - getche()
  - putch()

Reading and writing a single character

```
#include<stdio.h>
void main()
{
    char ch1,ch2;
    printf("Enter two characters");
    ch1=getchar();
    ch2=getchar();
    printf("\nthe two characters are :");
    putchar(ch1);
    printf("\t");
    putchar(ch2);
}
```

Example of getch, getche, and putch

```
#include<stdio.h>
void main()
{
	char ch1,ch2;
	printf("Enter first character");
	ch1=getche();
	printf("\nEnter second character");
	ch2=getch();
	printf("\nthe first character is :");
	putch(ch1);
	printf("\nthe second character is :");
	putchar(ch2);
}
```



```
Enter first charactera
Enter second character
the first character is :a
the second character is :b_
```