**printf(char \*format, arg1, arg2, …)**

This function prints the character on standard output and returns the number of character printed the format is a string starting with **%** and ends with conversion character (like c, i, f, d, etc.).  
Between both, there can be elements governing the printing format. Below is its description

1. A minus(-) sign tells left alignment.
2. A number after **%** specifies the minimum field width to be printed if the characters are less than the size of width the remaining space is filled with space and if it is greater than it printed as it is without truncation.
3. A period( . ) symbol separate field width with the precision.

Precision tells the maximum number of digits in integer, characters in string and number of digits after decimal part in floating value.

**Character format specifier : %c**

|  |
| --- |
| #include <stdio.h>  int main()  {      char ch = 'A', xy=’b’;      printf("%c\n", ch);  printf(“the value of ch is %c and value of xy is %c\n”,ch,xy);  the value of ch is A and value of xy is b      return 0;  } |

**Output:**

A

**Integer format specifier : %d, %i**

|  |
| --- |
| #include <stdio.h>  int main()  {      int x = 45, y = 90;      printf("%d ", x);      printf("%i\n", x);      return 0;  }  4545 |

**Output:**

45

45

**Floating-point format specifier : %f, %e or %E**

|  |
| --- |
| #include <stdio.h>  int main()  {      float a = 12.67;      printf("%f\n", a);      printf("%e\n", a);      return 0;  } |

**Output:**

12.670000

1.267000e+01

**Unsigned Octal number for integer : %o**

|  |
| --- |
| #include <stdio.h>  int main()  {      int a = 67;      printf("%o\n", a);      return 0;  } |

**Output:**

103

**Unsigned Hexadecimal for integer : %x, %X**

|  |
| --- |
| #include <stdio.h>  int main()  {      int a = 15;      printf("%x\n", a);      return 0;  } |

**Output:**

f

**String printing : %s**

|  |
| --- |
| #include <stdio.h>  int main()  {      char a[] = "VIT";      printf("%s\n", a);      return 0;  } |

**Output:**

VIT

**More formatting**

|  |
| --- |
| #include <stdio.h>  int main()  {      char str[] = "ABCDExyzABCDE";      printf("%20s\n", str);      printf("%-20s\n", str);      printf("%20.5s\n", str);      printf("%-20.5s\n", str);      return 0;  } |

**Output:**

ABCDExyzABCDE

ABCDExyzABCDE

ABCDE

ABCDE

**scanf(char \*format, arg1, arg2, …)**

This function take input using standard input (keyboard) and store it in variable accordingly. It returns the number of items successfully read. Formal parameter arg1, agr2, .. must be a pointer

|  |
| --- |
| #include <stdio.h>  int main()  {      int a = 0; int a;      scanf("%d", &a); // input is 45      printf("%d\n", a);      return 0;  } |

**Integer may be octal or in hexadecimal : %i**

|  |
| --- |
| #include <stdio.h>  int main()  {      int a = 0;      scanf("%i", &a); // input is 017 (octal of 15 )      printf("%d\n", a);      scanf("%i", &a); // input is 0xf (hexadecimal of 15 )      printf("%d\n", a);      return 0;  } |

**Double floating-point number : %lf**

|  |
| --- |
| #include <stdio.h>  int main()  {      double a = 0.0;      scanf("%lf", &a); // input is 45.65      printf("%lf\n", a);  printf(“%0.1f\n”,a);      return 0;  } |

**Output:**

45.650000

**String input : %s**

|  |
| --- |
| #include <stdio.h>  int main()  {      char str[20];      scanf("%s", str); // input is VIT      printf("%s\n", str);      return 0;  } |

**Output:**

VIT

**Character input : %c**

|  |
| --- |
| #include <stdio.h>  int main()  {      char ch;      scanf("%c", &ch); // input is A      printf("%c\n", ch);      return 0;  } |

Many other format specifiers are also there  
1.%u for an unsigned integer.  
2.%lld for long long int.  
3.%o octal integer without leading zero  
4.%x hexadecimal integer without 0x before the number.