**Input Output**

* Scansets – represented by %[] , inside square brackets we can specify single characters or range of characters and scanf process only those characters which are part of scanset.

If first character of scanset is ‘^’, then the specifier will stop reading after first occurrence of that character.

* Puts() moves the cursor to next line after printing the string.
* Fputs(str,stdout) – will print str and don’t move the cursor in next line.
* If % sympol is used then printf will give error but puts will not give error.
* %n format specifier - printf(“ hii this is &n Aditya”,&a);

Hence there are 12 charcters before %n , 12 will be store in a as a value.

And it will print 12 if you print a.

* Sprint(buffer,str) – it will store str in buffer and then print buffer to print str.
* %d specify single decimal integer while %i specify integer.
* %d prints the value as it is but %i print the decimal value and not in octal or hexadecimal value.
* Buffer – a temporary storage area .
* While taking the input of string the buffer does not get cleared for the next input and consider the previous input for the next also. So to clear this issue –

1. Fflush(stdin) –fflush(stdin)is used to clear out the buffer or stream. It is placed after scanf.
2. while ((getchar()) != '\n'); after scanf .

* scanf(“%\*s %d”,&a); - it will ignore all the string until the next space or newline and read only integer value.
* Scanf(“%\*d %s”,&a); - it will ignore all the integer values until the next space or newline and store only string values.
* getchar\_unlocked() is similar to getchar() with the exception that it is not thread safe. Faster than getchar().
* Similarly, there are getc\_unlocked() putc\_unlocked(), and putchar\_unlocked() which are non-thread-safe versions of getc(), putc() and putchar() respectively.
* Always put \n or space in scanf to clear out the buffer to so that buffer is available for next input and no unexpected output generated.
* The i**sprint()** function checks whether a character is a printable character or not. Return true or false value.
* The **iscntrl()** function is used to checks whether a character is a control character or not. Return true or false value.
* rand() function is used in C to generate random numbers.
* **RAND\_MAX:** is a constant whose default value may vary

between implementations but it is granted to be at least 32767.

* The srand() function sets the starting point for producing a series of pseudo-random integers.
* Just write srand(time(0)); at the starting of the program and then the rand() finction always generate different set of numbers when you compile .
* **srand() — Set Seed for rand() Function**.