classmate
Date
Page

Input Output in C

21) Input & Output in C

→ Most languages use the concept of Buffer
→ Data is taken in the form of chunks.
→ Process is a program in execution.
stdin → Standard input
stdout → Standard output
stderr → Standard error

→ Input func ⇒ scanf(), fgets() → reading a string (include space), getchar() → read a character....

→ Output func → printf(), puts() → strings, fputs(), putchar() → returning a string on output device using stdout as parameter

22) printf() in C

→ Syntax

(i) printf("....."); %d = int
(ii) printf("...", ...); %c = char
(iii) print new line. %u → unsigned int.

→ %d with characters is their respective ASCII values.

* "\n" is used to read the input until enter is pressed.

23) scanf() in C

Syntax

scanf("....", ...)

"...." → format string

.... → comma seperated list of address of variables

Eg:- scanf("%d %f %c", &a, &b, &c);

→ We have place holders in scanf().

→ (&) gives the address of the variables.

Eg:-
```
int main(){
    char name[50];
    printf("enter your name: \n");
    scanf("%s", name); → no (&) used here.
    printf("Hello %s, \n", name);
```

→ scanf() takes in input untill a (space) is encountered

→ fscanf() helps to read info from a file.

24) fgets() in C

→ It is used to read a string with spaces

Syntax

fgets(stringName, maxlengthofstring, stdin);

→ stdin is a filestream

→ The length of a string is always terminated with (10).

→ To prevent program crash, the limit is used in fgets().

→ The maxLengthOfString = arraySize
```
int main(){
    char name[50];
    printf("enter your name");
    fgets(name, 50, stdin);
printf("Hi %s, \n", name);
}
```

25) Format specifiers for integers

%d = int
%u = unsigned int
ld = long
lld = long long
zd = size_t
x = Hexadecimal
o = octal
i = almost same as d

```
int main(){
int x = 10;
printf("%d \n", x);
long y = 20;
printf("%ld \n", y);
unsigned z = 30;
printf("%u \n", z);
long long w = 40;
printf("%lld \n", w);
size_t u = sizeof(x);
printf("%zd", u);
}
```

26) Format specifiers for floating point

→ Floating point nos are automatically upgraded to double before printing.

→ In fixed point notation, we have 6 decimal places.

%f = double in fixed point notation
%e = double in e form
%g = double in fixed/e form
- → 7 of all max 6 digits
- → No trailing 0's.
- → Keep trailing 0's (%#g).
- → Can upgrade to %e auto.

```
int main(){
float x = 10.25;
double y = 125.3e7648;
double z = 15e+6;
printf("%f \n", x);
printf("%e \n", y);
printf("%g \n", z);
```

Output

10.250000
1.253e7648
1.5e+07

27) Other format specifiers

%c = char
%s = String
%p = pointer (or addresses)
→ use (&) symbol
→ give memory address in hexadecimal
→ depends on system as well

```
int main(){
int a = 'g';
char b[] = "gfg";
printf("%c %s %p",
    a, b, &a);
```

Output
g gfg 0x7ffc6a2fb5f

%n = stores no of characs printed so far before encountering %n.

28) Width & Precision in printf()
→ Print the formulated data
→ Width → minimum characters to print on screen.
Syntax
printf("%5d\n", x); OR printf("%5s\n", y);
→ spaces are printed if the no of characters < minimum.
→ For float nos. point is also considered as characters.

Flag ships with width
(-) minus → left aligned the characters & spaces are on right side.
(0) Zero flag → rly for numeric value. The spaces are replaced with (0s).
(+) plus flag → The (+ve) nos. are printed with positive sign.

Eg:

```
int main(){
  int x = 124;
printf("%d\n", x);
printf("%5d\n" x);
  char y[] = "gfg";
printf("%5s\n", y);
  float z = 1.2;
printf("%5g\n", z);
  return 0;
}
```

```
int main(){
  int x = 124
  char y[] = "gfg";
  float z = 1.2;
printf("%-5d%-5s%-f\n", x, y, z);
  printf("%05d\n", x);
  printf("%+5d", x);
}
```

Output

```
124   gfg   1.2
00124
+124
```

Precision

→ It is specified after the (.)

→ Mainly used for strings & floating point nos. to get the decimal places correct.

→ We can use flag, width & precision together.

```
int main(){
char x[] = "geeksforgeeks";
printf("%.3s\n", x);
printf("%5.3s\n", x);
printf("%.*s\n", 5, 3, x);
printf("%-5.3s\n", x);
  return 0;
}
```

Output

```
gee
  gee
  gee
gee
```

```
int main(){
  double x = 1.345;
printf("%.2f\n", x);
printf("%5.2f\n", x);
printf("%.*f\n", 5, 2, x);
printf("%-5.2f\n", x);
  return 0;
}
```

Output

```
1.34
 1.34
01.34
1.34
```

30) Escape sequence in C

→ Eg: print a double quotes, we will use escape sequences

→ Syntax:

"\" ..... \""

Eg: printf("Welcome to \"Bengaluru\"");

→ Here in (1) together form an escape sequence

→ (\n) is used to go to next new line.

→ (\\) two back slashes tell that there is to be a back slash printed.