UNIT – 3
Input and Output

C PROGRAMMING (CSC115)
BSC CSIT FIRST SEMESTER (TU)

PREPARED BY:

DABBAL SINGH MAHARA

ASCOL (2025)

Data Input and Output

- Generally, a program reads input data from keyboard (standard input device) then processes the input data and the result is displayed on the screen or monitor (standard output device).
- Thus, reading the data from input devices and displaying the result on the screen are the two main functions of any program.
- These functions are known as input output operations.

Data Input and Output…

- To perform input and output operations, C provides a number of built-in input and output functions.
- Input functions take data from input devices and transfer into the program variables.
- Similarly, output functions send the results from program to output devices.

Standard input and output functions

- Keyboard is a standard input device, the input functions used to read data from keyboard are called standard input functions.
  - scanf(), getchar(), getch(), getche(), gets() etc are standard input functions.
- The functions which are used to display result on the screen are called standard output functions.
  - The standard output functions are: printf(), putchar(), putch(), puts() etc.

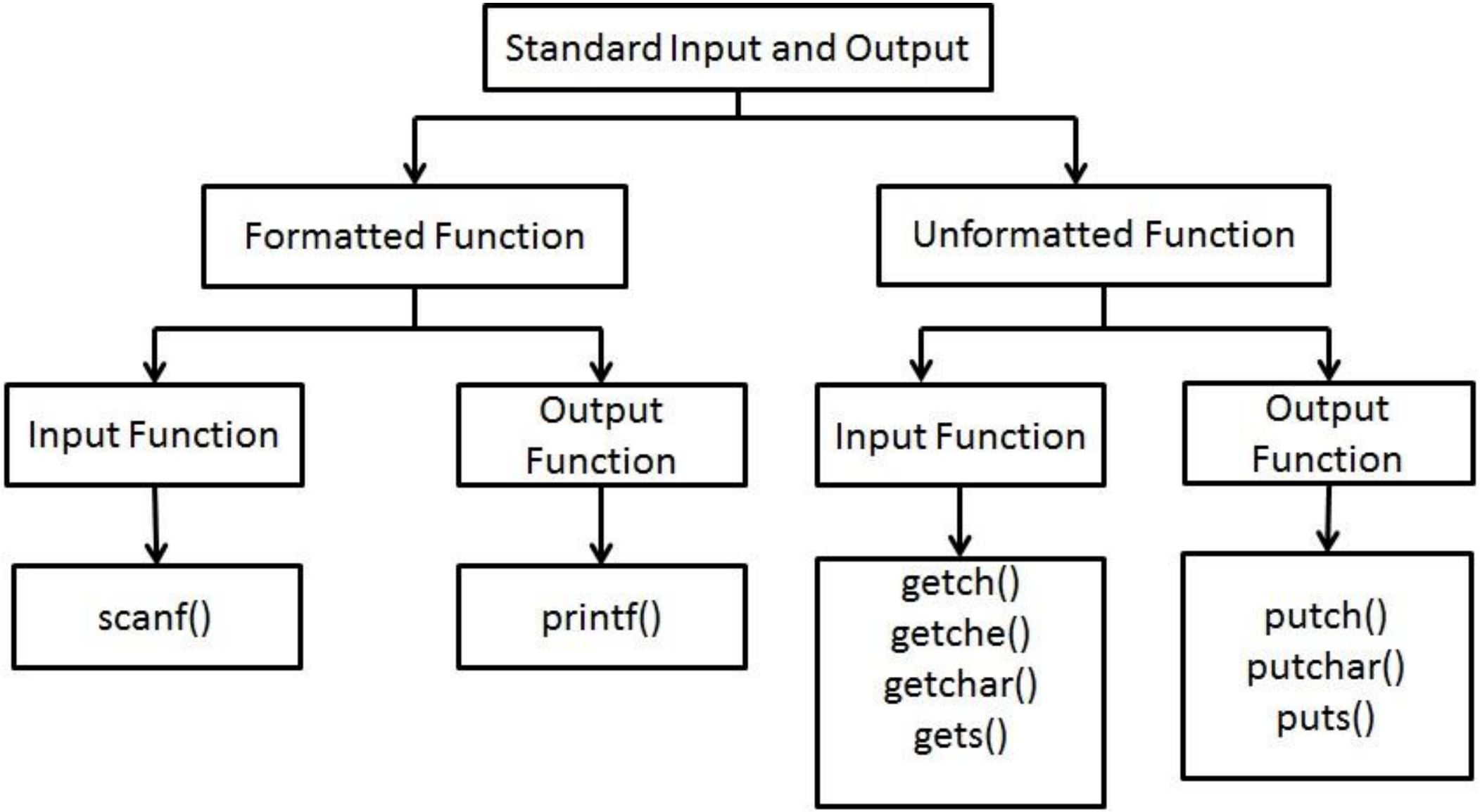
In C, the standard library, <stdio.h> provides functions for input and output.

I/O Functions in C

- I/O functions are classified into two types

1. Formatted Functions: These are the functions that are used to format input output data as per user requirements.eg: scanf() and printf()

2. Unformatted Functions: These functions do not allow to read or display in desired format. Examples: getchar(), putchar(), getch(), putch(), gets(), puts()


Standard Input and Output
Formatted Function
Unformatted Function
Input Function
Output Function
Input Function
Output Function
scanf()
printf()
getch()
getche()
getchar()
gets()
putch()
putchar()
puts()

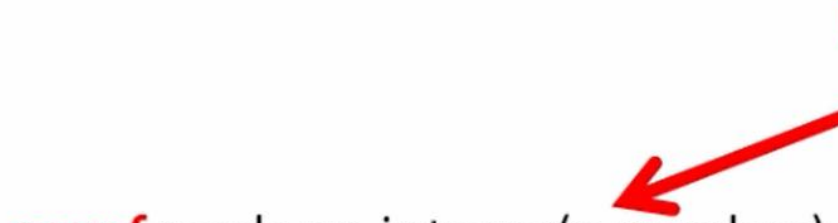
Formatted Input

- Formatted input refers to an input data that has been arranged in a particular format.
- The built-in function scanf() can be used to enter input data into the computer from a standard input device.
- The general form of scanf() is,

  scanf("control string", arg1, arg2, …, argN);
  - The control string refers to the field format in which the data is to be entered.
  - The arguments arg1, arg2, …., argN specify the variables where data is stored. Generally, there is & to denote the address of variables.

Example: Scanf() function

```
int age ;
printf("Enter your age : ");
scanf("%d", &age);
```



scanf reads an integer(a number) which the user enters

scanf puts that read value "At the address of" 'age' variable

Formatted Input…

- The general format of a control string is:

[whitespace character][ordinary character]%[field width][conversion character]

- **Whitespace character:** is optional part. A whitespace character in control string means that it reads data without storing consecutive white space character till the next non whitespace character in the input.
- **Ordinary characters:** are included to take input in a certain pattern.
- **Field width:** specifies the maximum number of characters to be read. The input data may contain fewer characters than the specified field width but characters in actual input data cannot exceed the specified field width.
- **Conversion characters:** The conversion character is used on the basis of data type of variable to store.

Conversion Character/ Format Scecifier

Data type		Format specifier
Integer	short signed	%d or %I
	short unsigned	%u
	long singed	%ld
	long unsigned	%lu
	unsigned hexadecimal	%x
	unsigned octal	%o
Real	float	%f
	double	%lf
Character	signed character	%c
	unsigned character	%c
String		%s

```
#include <stdio.h>
#include <conio.h>
int main()
{
int day, month, year;
printf("Enter day, month, year in DD-MM-YYYY format:");
scanf(" %d-%d-%d", &day, &month, &year);
printf("\nDay: %d \t Month: %d \t Year:%d", day, month, year);
getch();
return 0;
}
```

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int n;
        printf("Enter a number of max 5 digits:");
        scanf("%5d",&n);
        printf("Number: %d",n);
        getch();
        return 0;
}
```

```
#include<stdio.h>
Int main()
{
    int n;
    char ch;
    printf("Enter a number" );
    scanf("%d" , &n);
    printf("Enter a character ");
    scanf(" %c", &ch);
    printf("\n Number = %d \t Character = %c", n, ch);
    return 0;
}
```

Formatted Output

- Formatted output refers to the output of data that has been arranged in a particular format.
- The built-in function printf() is used to output data from the computer onto a standard output device i.e. screen.
- The general form of printf() is,

  printf("control string", arg1, arg2, …., argN);

Formatted Output…

- The control string has the form,
- %[flag][field width][.precision]conversion character

- **Flags**
  - Flags may be
    - - (left justified)
    - + (sign precedes)
    - 0 (leading 0s)
    - blank space (preceding space)
    - # (causes octal and hex items to be preceded by 0 and 0x)

Formatted Output…

- **Field Width**
  - The field width is an integer specifying the minimum output field width.
  - If the no. of characters in the corresponding data item is less than the specified field width, then the data item will be preceded by enough leading blanks to fill the specified field.
  - If the no. of characters in the data item exceeds the specified field width, then additional space will be allocated to the data item so that the entire data item will be displayed.

Formatted Output…

- **Field Width…**
  - Any integer no. can be displayed in a desired width using the format, %wd

    where w is the minimum field width for the output and d specifies that the value is an integer.

E.g.

```
int n=1234;
printf("%d", n);
```

Formatted Output…

printf("%6d", n);

printf("%2d", n);

printf("%-6d", n);

printf("%06d",n);

```
#include <stdio.h>
#include <conio.h>
Int main()
{
int a=10;
float x=11.123456;

printf("%-6d",a);
printf("\n");
printf("%+d",a);
printf("\n");
printf("%09d",a);
printf("\n");
printf("% d",a);
printf("\n");
printf("%#o",a);
printf("\n");
printf("%#0x",a);
printf("\n");
printf("%7.2f",x);
getch();
}
```

//precision format %w.pf

Unformatted Functions

- Unformatted functions do not allow the user to read or display data in desired format.
- The functions getchar(), putchar(), gets(), puts(), getch(), getche(), putch() are unformatted functions.

getchar() and putchar()

- The getchar() function reads a character from a standard input device. The syntax is,

  char_variable = getchar();

  where char_variable is a valid C identifier. When this statement is encountered, the computer waits until a key is pressed and then assigns this character to char_variable.

- The putchar() function displays a character to the standard output device. The syntax is,

  putchar(char_variable);

  where char_variable is a char type variable containing a character.

```
#include <stdio.h>
#include <conio.h>
int main()
{
   char gender;

   printf("Enter your gender (M or F):");
   gender = getchar();
   printf("Your gender is:");
   putchar(gender);
   getch();
}
```


≡ printf(“%c”, gender)

getch(), getche() and putch()

- The functions getch() and getche() reads a single character the instant it is typed without waiting for the Enter key to be hit.
- The difference between getch() and getche() is that getch() reads a character typed without echoing it on the screen, while getche() reads the character and echoes it on the screen.
- Syntax for getch(),

```
char_variable = getch();
```

- Syntax for getche(),

```
char_variable = getche();
```

- The function putch() prints a character onto the screen. Its syntax is,

  putch(char_variable);

- Since getch(), getche() and putch() are defined under the standard library functions in <conio.h>, this must be included in our program.

```
#include <stdio.h>
#include <conio.h>
int main()
{
char ch1,ch2;
printf("Enter 1st character:");
ch1=getch();
printf("\nEnter 2nd character:");
ch2=getche();
printf("\n 1st character:");
putch(ch1);
printf("\n2nd character:");
putch(ch2);
getch();
return 0;
}
```

gets() and puts():

- gets(): This function is used to read a string containing whitespaces until a newline character is encountered.
- This function is alternative to scanf() for reading string.
- Unlike scanf() function, it does not skip whitespaces.
- Syntax: gets(string_variable);
- puts(): this function is **used to display the string** on the screen.
- Syntax: puts(string_variable);

Example

```
// gets() and puts(): string handling functions
#include<stdio.h>

int main()
{
        char name[25];  // name is a string variable to your name
        printf("\n Enter your name :");
        gets(name);

        printf("\n Your name is:”);
        puts(name);
        return 0;
}
```

Homework

1. What do you mean by input/output in computer system?
2. What are the formatted and unformatted input/output functions?
3. Write and explain the syntax of printf() and scanf() functions.
4. What is difference between scanf() and gets()?
5. What is difference between getch() and getche()?

Thank you !