UNIT 3
INPUT AND OUTPUT

LH – 2HRS

PRESENTED BY:

ER. SHARAT MAHARJAN

C PROGRAMMING

CONTENTS (LH – 2HRS)

3.1 Input/Output Operation,

3.2 Formatted I/O (scanf, printf),

3.3 Unformatted I/O (getch-putch, getche, getchar-putchar and gets-puts

- Computer is an electronic device which takes data as input and performs processing to generate output.
- A computer program takes input generally from keyboard (standard input device), then processes it and sends the result to an output device (i.e. computer screen or monitor or data file).

3.1 Input/Output Operation

- Input is a process of reading data from input devices into program. C provides a set of built-in functions to read given input and feed it into the program as per requirement.
- Output is a process of displaying/writing data on screen, printer or in any file. C provides a set of built-in functions to output required data.
- As keyboard is a standard input device, the input functions used to read data from keyboard are called standard input functions. The **standard input functions are scanf(), getchar(), getche(), getch(), gets() etc.**
- Similarly, the output functions which are used to display the result on the screen are called standard output functions. The standard output functions are **printf(), putchar(), putch(), puts() etc.**
- In C, the header file stdio.h provides functions for input and output operations.

The input/output functions are classified into two types –

1. **Formatted functions**
2. **Unformatted functions**

3.2 Formatted I/O

- The functions which **allow input or output data to format according to user's requirement** are known as formatted I/O functions.
- The **input function scanf() and output function printf() are formatted I/O functions.**
- For example: the formatted functions can be used to specify the number of **digits to be displayed after decimal point**, **number of spaces before the data item and the position where the output is to be displayed.**

1. Formatted Input (scanf())

- The **built-in function scanf()** can be used to input data into the computer from a standard input device. The function can be used to input a numerical value, single character or string.
- 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 address of locations where the data is stored. Generally, **arguments are preceded by ampersand (&) to denote memory address**. The control string again contains multiple parts. The **general form** of a control string is:

  [whitespace_character] [ordinary_character] %[field_width] conversion_character

a. **Whitespace Characters [Optional]**

A whitespace character in control string means that it **reads data without storing the consecutive whitespace characters till the next non-whitespace character in the input**. **(inputting second character)**

b. **Ordinary Characters [Optional]**

They are included to **take input in a certain pattern. (date)**

c. **Field Width [Optional]**

It specifies the **maximum number of characters to be read**. The input data may contain fewer characters than specified field width but number of characters in the actual input data item can't exceed the specified field width. **Any characters that exceeds the specified field width will be skipped out. (marks)**

d. **Conversion Character**

The conversion character is used on the basis of type of variable or data item to be input. The commonly used conversion characters are: **%c, %d, %f, %s etc.**

The **scanf()** function supports the following conversion specifications for strings –

%[characters]

%[^characters]

- The specification **%[characters] means that only the characters specified within the brackets are allowed** in the input of string. **If the input string contains any other character, the reading of string will be terminated at the first encounter** of such a character.
- In specification **%[^characters], the characters specified after the caret (^) are not allowed in the string and reading will be terminated.**

2. Formatted Output (printf())

- Formatted output functions are used to display or store data in a particular specified format. The printf() is an example of formatted output function.
- The **general form of printf()** statement is:

  printf("control string", arg1, arg2, … , argn);

- The control string may consist any of the following data items:

I. Characters that will be printed on the screen as they appear.

II. Format specifications that define the output format for display of each item.

III. Escape sequence characters such as \n and \t.

IV. Any combination of characters, format specifications and escape sequences.

- The arguments arg1, arg2, … , argn are the variables whose values are formatted and printed according to the specifications of the control string. The control string has the form:

  %[flag] [field width] [.precision] conversion character

1. Flags [Optional]

The flags may be -, +, 0 or #. A '-' flag is used to indicate data item to be left-justified. A '+' flag is used to display sign (either positive or negative) to precede each signed numerical data item. A '0' flag is used to indicate leading 0s to appear instead of leading blanks in right justified data item whose minimum width is larger than the data item. The flag '#' is used with %o or %x to indicate octal and hexadecimal items to be preceded by 0 or 0x respectively. Similarly, the flag '#' is used with %e, %f or %g to enforce a decimal point in floating point numbers, even if it is a whole number.

2. Field Width [Optional]

The field width is an integer which specifies the minimum number of characters or digits to be displayed in output operation. If the number 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 number of characters in the data item exceeds the specified field width, then the entire data item will be displayed.

3. Precision [Optional]

- The precision is expressed in integer which specifies number of digits to be displayed after decimal point in floating point number. It begins with period (.).

4. Conversion Character

- The conversion character for printf() is similar to that of scanf(). The conversion character depends upon the type of the variable or constant to be displayed. The commonly used printf() conversion characters are %c, %d, %f, %s etc.

3.3 Unformatted I/O

- Unformatted I/O functions do not allow to read or display data in desired format. These type of library functions basically deal with a single character or a string of characters. The functions getchar(), putchar(), gets(), puts(), getch(), getche(), putch() are some examples of unformatted functions.

1. <u>getchar() and putchar()</u>

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

  Syntax: character_variable = getchar();

The getchar() function makes wait until a key is pressed and then assigns this character to character_variable.

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

  Syntax: putchar(character_variable);

2. getch(), getche() and putch()

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

Syntax:

```
character_variable = getch();
character_variable = getche();
```

- In both functions, the character typed is assigned to the **char** type variable character_variable.
- The function **putch**() prints a character onto the screen.

3. gets() and puts()

- The gets() function is used to read a string of text, containing whitespaces, until a newline character is encountered. It can be used as alternative function for scanf() function for reading strings. Unlike scanf() function it doesn't skip whitespaces (i.e. it can be used to read multi words string).

  Syntax: gets(string_variable);

- The puts() function is used to display the string on the screen.

  Syntax: **puts(string_variable);**

THANK YOU FOR YOUR ATTENTION