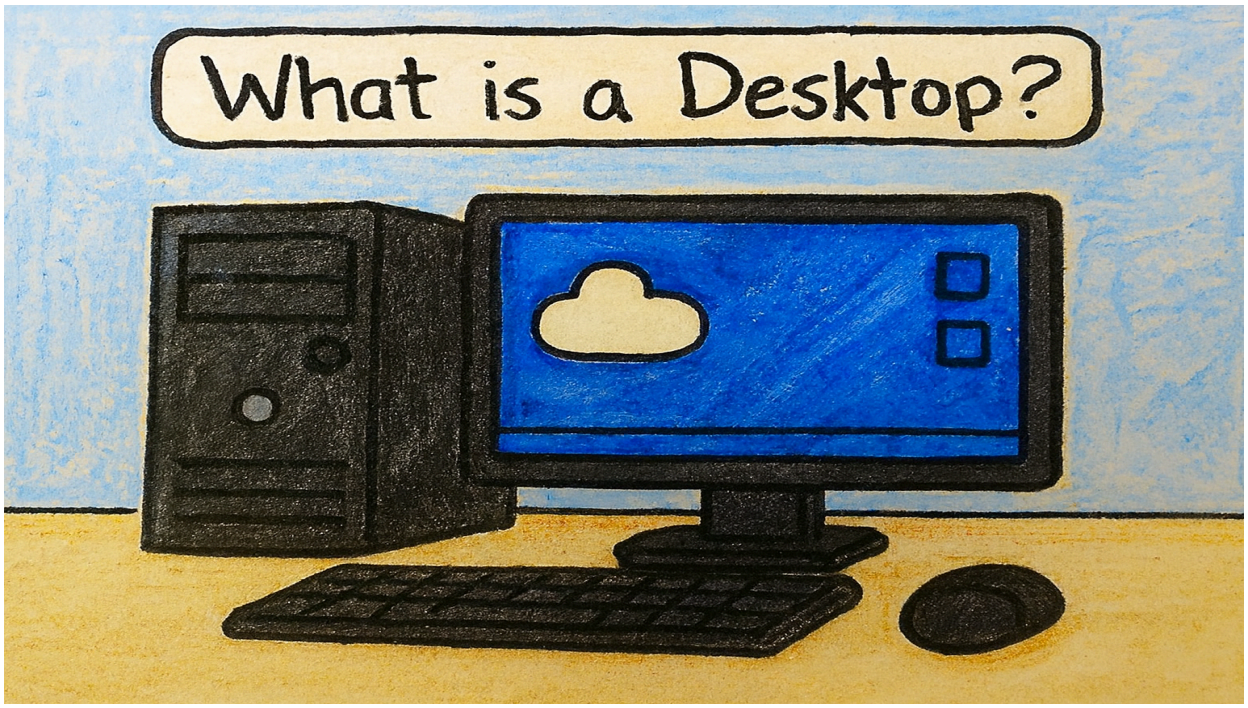

Class: 12th

Subject: Computer

Chapter 10: INPUT/OUTPUT

📍 Important MCQs:

1. Which library provides standard input and output functions in C?

(a) stdlib.h

(b) conio.h

(c) stdio.h ✅

(d) math.h

2. In C, standard input and standard output refer to:

(a) File and printer

(b) Mouse and keyboard

(c) Keyboard and monitor ✅

(d) Scanner and file

3. Which function is used for formatted output in C?

(a) scanf()

(b) printf() ✅

(c) input()

(d) output()

4. The first argument of the printf() function is always a:

(a) Variable

(b) Constant

(c) Format string ✅

(d) Expression

5. The format string in printf() is enclosed in:

(a) Single quotes

(b) Curly brackets

(c) Double quotes ✅

(d) Square brackets

6. Which symbol is used to begin a format specifier in C?

(a) #

(b) &

(c) % ✅

(d) $

7. Which format specifier is used to print an integer value?

(a) %f

(b) %c

(c) %d ✅

(d) %s

8. Which format specifier is used for float data type?

(a) %d

(b) %lf

(c) %f ✅

(d) %e

9. Which format specifier is used for double data type?

(a) %f

(b) %lf ✅

(c) %d

(d) %g

10. Which format specifier is used to display a single character?

(a) %s

(b) %c ✅

(c) %d

(d) %u

11. Which format specifier is used to print a string?

(a) %c

(b) %d

(c) %s ✅

(d) %f

12. Which format specifier is used for unsigned integers?

(a) %d

(b) %u ✅

(c) %x

(d) %o

13. Field-width in C refers to:

(a) Data type size

(b) Memory location

(c) Number of columns used to display a value ✅

(d) Number of variables

14. In the statement printf("Area = %4d", area);, the number 4 represents:

(a) Number of variables

(b) Number of decimal places

(c) Field-width (columns) ✅

(d) Number of lines

15. If the specified field-width is smaller than the value, C will:

(a) Generate an error

(b) Truncate the value

(c) Expand the field-width automatically ✅

(d) Stop execution

16. In the format specifier %m.nf, the value of n represents:

(a) Total field-width

(b) Integer part

(c) Number of decimal places ✅

(d) Exponential form

17. Which escape sequence is used to move the cursor to the next line?

(a) \t

(b) \b

(c) \n ✅

(d) \f

18. Which escape sequence is used to insert a tab space?

(a) \n

(b) \t ✅

(c) \b

(d) \r

19. Which escape sequence is used to print a double quote on the screen?

(a) ""

(b) '

(c) " ✅

(d) \

20. All escape sequences in C begin with:

(a) %

(b) #

(c) /

(d) \ ✅

21. Which C function is used to accept input from the user during program execution?

(a) printf()

(b) getch()

(c) scanf() ✅

(d) putchar()

22. The scanf() function belongs to which header file?

(a) conio.h

(b) stdlib.h

(c) stdio.h ✅

(d) string.h

23. Which one is the correct syntax of the scanf() function?

(a) scanf(var1, var2);

(b) scanf("format string", var1, var2);

(c) scanf(format string, &var1, &var2);

(d) scanf("format string", &var1, &var2); ✅

24. In scanf(), the ampersand (&) symbol is used to:

(a) Declare a variable

(b) Print the variable

(c) Pass the address of a variable ✅

(d) Convert data type

25. If the ampersand (&) is not used with a variable in scanf(), the program will:

(a) Work correctly

(b) Give compile-time error

(c) Store zero in variable

(d) Store a garbage value ✅

26. The format string of scanf() consists of:

(a) Text and variables

(b) Only text

(c) Only format specifiers ✅

(d) Expressions and constants

27. Which format specifier is used to input a double value using scanf()?

(a) %f

(b) %d

(c) %lf ✅

(d) %e

28. Which function is used to input a single character without pressing the ENTER key?

(a) scanf()

(b) getchar()

(c) getch() ✅

(d) printf()

29. Which function echoes the typed character on the screen?

(a) getch()

(b) getche() ✅

(c) scanf()

(d) putchar()

30. The functions getch() and getche() are declared in:

(a) stdio.h

(b) stdlib.h

(c) string.h

(d) conio.h ✅

📍 Important Short Questions:

1. What is meant by standard input and standard output in C?

Answer:

👉 In C language, standard input refers to the keyboard, which is used to enter data into the program, while standard output refers to the monitor/screen, which is used to display the results of a program.

Example:

Data is entered using keyboard and results are shown on the screen.

2. Name the two standard input/output functions used in C.

Answer:

👉 The two standard input/output functions used in C are printf() and scanf().

👉 **printf()** is used to display output, while **scanf()** is used to take input from the user.

3. Which header file is required to use printf() and scanf()?

Answer:

👉 To use printf() and scanf() functions, the stdio.h header file must be included in the program.

Example:

#include <stdio.h>

4. What is printf() function?

Answer:

👉 The printf() function is a standard library function used to display formatted output on the screen according to the given format string.

5. What is a format string in printf()?

Answer:

👉 A format string is a character string enclosed in double quotation marks that tells the printf() function what to print and how to print it.

Example:

"Area of Square = %d"

6. What is a format specifier?

Answer:

👉 A format specifier is a special symbol starting with (%) that specifies the data type and format of a variable to be displayed.

Example:

%d for integer, %f for float

7. Write the format specifier for integer data type.

Answer:

👉 The format specifier used for an integer data type is %d.

8. Write the format specifier for floating point data type.

Answer:

👉 The format specifier used for floating point (float) data type is %f.

9. What is field-width in C?

Answer:

👉 Field-width refers to the number of columns reserved on the screen to display a value using the printf() function.

10. How is field-width specified for integers in printf()?

Answer:

👉 Field-width is specified by writing a number between % and d in the format specifier.

Example:

%4d ⟶ reserves 4 columns for an integer value.

11. What happens if the specified field-width is smaller than the value?

Answer:

👉 If the specified field-width is smaller than the value, C automatically expands the field-width so that the complete value can be displayed.

12. Write the general form of floating point format specifier.

Answer:

👉 The general form of a floating point format specifier is %m.nf,

where m is the total field-width and n is the number of decimal places.

Example:

%6.2f

13. What is an escape sequence?

Answer:

👉 An escape sequence is a special character combination that starts with a backslash (\) and is used to perform special formatting actions in output.

14. Why are escape sequences used to display single and double quotes?

Answer:

👉 Single and double quotes are part of string delimiters, so escape sequences are used to print them as characters on the screen without causing errors.

15. Write any two escape sequences with their functions.

Answer:

👉 \n ⟶ Moves the cursor to the next line

👉 \t ⟶ Inserts a horizontal tab space

16. What is meant by an interactive program?

Answer:

👉 An interactive program is a program that accepts input from the user during execution and displays output accordingly, making the program interactive.

Example:

A program that converts kilometers to meters by asking the user to enter a value.

17. Define the scanf() function in C.

Answer:

👉 The scanf() function is used to accept input from the user. It can read numeric values, characters, or strings.

Example:

scanf("%d", &num); reads an integer from the user.

18. Write the general syntax of the scanf() function.

Answer:

👉 The general syntax is:

scanf("format string", &var1, &var2, ...);

Example:

scanf("%lf", &kilometer); reads a double value into the variable kilometer.

19. What is the purpose of the format string in scanf()?

Answer:

👉 The format string specifies the type of data that is to be read from the user.

It contains only format specifiers, no text or constants.

Example:

"%d" ⟶ integer, "%f" ⟶ float

20. Why is the ampersand (&) operator used with variables in scanf()?

Answer:

👉 The ampersand & gives the address of the variable to scanf() so that the input value can be stored in the correct memory location.

21. What will happen if the ampersand (&) is omitted in scanf()?

Answer:

👉 If & is omitted, scanf() cannot locate the variable in memory and will store a garbage value, giving incorrect results.

22. Which format specifier is used to input a double value using scanf()?

Answer:

👉 The format specifier for double is %lf.

Example:

scanf("%lf", &kilometer);

23. Why is scanf() not suitable for character input in some situations?

Answer:

👉 Because scanf() requires pressing the ENTER key after typing a character, it is not suitable for real-time character input like in games or interactive menus.

24. What is the getch() function?

Answer:

👉 The getch() function reads a single character from the keyboard without waiting for the ENTER key.

It does not display the typed character on the screen.

Header file: conio.h

25. Differentiate between getch() and getche() functions.

Answer:

Function	Display typed character	Requires ENTER key	Description
getch()	No	No	Reads a character without echoing it.
getche()	Yes	No	Reads a character and immediately displays it.

Example:

- ch = getch(); ⟶ character stored but not shown
- ch = getche(); ⟶ character stored and shown immediately

📍 Exercise 10c

1. Fill in the blanks:

(i) The _____ function does not display characters on the output screen.

Answer: getch()

Explain:

👉 The getch() function reads a single character from the keyboard without displaying it on the screen.

Example:

ch = getch(); ⟶ character is stored but not shown on the screen.

(ii) _____ is an input function.

Answer: scanf()

Explain:

👉 The scanf() function is used to accept input from the user during program execution.

Example:

scanf("%d", &num); ⟶ reads an integer from the user.

(iii) %x is a format specifier for _____.

Answer: unsigned hexadecimal integers

Explain:

👉 %x is used to display or read numbers in hexadecimal format.

Example:

printf("%x", 255); ⟶ output: ff.

(iv) Escape sequences always begin with a _____.

Answer: backslash ()

Explain:

👉 An escape sequence is a *special character combination starting with * that performs formatting actions.

Example:

\n ⟶ moves cursor to next line, \t ⟶ inserts a tab.

(v) The printf function is defined in _____.

Answer: stdio.h

Explain:

👉 printf() is a standard library function defined in stdio.h header file, used for formatted output.

Example:

```
#include <stdio.h>

printf("Hello World");
```

(vi) The ASCII code for Escape key is _____.

Answer: 27

Explain:

👉 Each key on the keyboard has a unique ASCII code. The Escape (Esc) key has ASCII value 27.

(vii) The escape sequence _____ represents the carriage return.

Answer: \r

Explain:

👉 \r moves the cursor to the beginning of the current line without advancing to the next line.

Example:

printf("Hello\rWorld"); ⟶ World overwrites Hello.

(viii) There are total _____ columns on the output screen.

Answer: 80

Explain:

👉 Standard console screen has 80 columns for displaying characters horizontally.

Example: Each value printed occupies a column on the screen.

(ix) The symbol for address of operator is _____.

Answer: &

Explain:

👉 The address-of operator (&) is used in C to give the memory address of a variable.

Example:

scanf("%d", &num); ⟶ &num passes the address to store input.

(x) \dddd is used to print ASCII code in _____ notation.

Answer: octal

Explain:

👉 \dddd represents a character using its ASCII code in octal notation.

Example:

\101 ⟶ prints A (ASCII 65 in octal).

2. Choose the correct option:

(i) The function getche() is defined in:

(a) stdio.h

(b) string.h

(c) math.h

(d) conio.h ✅

(ii) The escape sequence for backslash is:

(a) /

(b) \b ✅

(c) //

(d) \t

(iii) The format specifier %u is used for:

(a) integer

(b) unsigned short ✅

(c) unsigned float

(d) unsigned long int

(iv) Which type of error occurs when a number exceeds storage limit?

(a) Arithmetic overflows ✅

(b) Arithmetic underflow

(c) Truncation

(d) Round off

(v) The symbol '=' represents:

(a) Comparison operator

(b) Assignment operator ✅

(c) Equal-to operator

(d) None of these

(vi) Which of the following operators has lowest precedence?

(a) !

(b) +

(c) = ✅

(d) ==

(vii) Relational operators are used to:

(a) Establish a relationship among variables

(b) Compare two values ✅

(c) Construct compound condition

(d) Perform arithmetic operations

(viii) C is a strongly typed language, this means that:

(a) Every program must be compiled before execution

(b) Every variable must be declared before it is being used ✅

(c) The variable declaration also defines the variable

(d) Sufficient data types are available to manipulate each type of data

(ix) The logical not operator, denoted by !, is a:

(a) Ternary operator

(b) Unary operator ✅

(c) Binary operator

(d) Bitwise operator

(x) a += b is equivalent to:

(a) b += a

(b) a = b

(c) a = a + b ✅

(d) b = b + a

3. Write T for true and F for false statement:

(i) printf and scanf are standard identifiers.

Answer: T ✅

(ii) In C language you must declare all variables before using them.

Answer: T ✅

(iii) Standard data types are not predefined in C language.

Answer: F ❌ (Correct: Standard data types are predefined like int, float, double, char)

(iv) The double data type requires 4 bytes in memory.

Answer: F ❌ (Correct: double usually requires 8 bytes)

🌟 **Q.4: What do we mean by standard input and output? Illustrate the use of printf() and scanf() functions.**

❖ **Answer:**

Standard Input and Output in C:

👉 In C programming, standard input refers to the device from which a program receives data. Usually, this is the keyboard.

👉 Standard output refers to the device where a program displays its results. Usually, this is the monitor.

To perform input and output operations in C, we use two main standard library functions:

1. printf() – used for output
2. scanf() – used for input

1. printf() Function:

- printf() is used to display data on the screen in a formatted way.
- **Syntax:**

C

```
printf("format string", variable1, variable2, ...);
```

- **Example:**

C

```
#include <stdio.h>

void main() {

   int marks = 80;

   printf("Marks obtained: %d", marks);

}
```

Output:

Marks obtained: 80

- **Here**, %d is a format specifier for integer, and marks is the variable whose value is displayed.

2. scanf() Function:

- scanf() is used to accept input from the user.
- It requires the address of the variable where the input is to be stored.

Syntax:

C

```
scanf("format string", &variable1, &variable2, ...);
```

- **Example:**

C

```
#include <stdio.h>

void main() {
    int marks;
    printf("Enter your marks: ");
    scanf("%d", &marks);
    printf("You entered: %d", marks);
}
```

Sample Input/Output:

```
Enter your marks: 85
```

You entered: 85

- **Here**, %d is used for integer input, and &marks provides the memory address of the variable marks.

✦ **Summary:**

- **Standard input:** keyboard
- **Standard output:** monitor
- printf() is used to display output, scanf() is used to accept input.
- Format specifiers like %d, %f, %c, %s define the type of data.

🌟 **Q.5: Illustrate the difference between format specifiers and field-width specifiers with examples.**

❖ **Answer:**

1. Format Specifiers:

Definition: Format specifiers are symbols used in printf() or scanf() to indicate the type of data a variable holds.

Purpose: They tell the compiler how to interpret and display the value of a variable.

Common Format Specifiers:

Symbol	Data Type
%d	int
%f	float
%lf	double
%c	Char
%s	String

Example:

C

```
#include <stdio.h>

void main() {

    int marks = 85;

    float percentage = 76.5;

    printf("Marks: %d\n", marks);

    printf("Percentage: %f\n", percentage);

}
```

Output:

Marks: 85

Percentage: 76.500000

- **Here**, %d is used for integers (marks) and %f is used for floating-point numbers (percentage).

2. Field-Width Specifiers:

- **Definition:** Field-width specifiers define how many columns (spaces) a value should occupy on the screen.
- **Purpose:** They are used to align output neatly.
- **Syntax:** Place a number between % and format specifier: %m.d
  - m ⟶ total width (number of spaces)
  - d ⟶ data type

Example with integers:

C

```
#include <stdio.h>

void main() {

   int a = 25, b = 5;

   printf("A = %4d\n", a);

   printf("B = %4d\n", b);

}
```

Output:

A = 25

B = 5

- **Here**, 4 is the field width, so each number occupies 4 spaces, right-justified.

Example with floating-point numbers:

C

```
#include <stdio.h>

void main() {

   float height = 15.245;

   printf("Height = %6.2f\n", height);

}
```

Output:

Height = 15.25

6.2 means total width = 6 columns, 2 digits after decimal.

✦ **Difference Summary:**

Feature	Format	Field-Width

	Specifier	Specifier
Purpose	Specifies data type	Specifies width and alignment of output
Example	%d, %f, %c, %s	%4d, %6.2f
Use	To interpret variable type	To format output layout on screen

🌟 **Q.6: Define the term 'escape sequence'. List names and uses of any five escape sequences.**

❖ **Definition:**

👉 In C, an escape sequence is a combination of characters that starts with a backslash () and represents a special action or character in the output.

👉 Escape sequences allow us to format output, insert tabs, new lines, backspace, or display special characters that cannot be typed directly.

General Form:

\ followed by a single character

Five Common Escape Sequences:

Escape Sequence	Use / Meaning
\n	Moves cursor to next line (newline)
\t	Inserts a tab space
\\	Prints a backslash \
\'	Prints a single quote ‘
\"	Prints a double quote “

Example Program:

C

```
#include <stdio.h>

void main() {

    printf("Hello\tWorld\n");

    printf("This is a backslash: \\\n");

    printf("Single quote: \' and Double quote: \"");

}
```

Output:

Hello	World

This is a backslash: \

Single quote: ' and Double quote: "

✦ **Summary:**

- Escape sequences **start with **.
- They are used to format text and display special characters.

Examples: \n, \t, \\, \', \".

🌟 **Q.7:**

a) Show the output displayed by the following program when the data entered are 10 and 15

Program (corrected version):

C

```
#include <stdio.h>

void main() {

   int m, n;

   printf("Enter two numbers (separated by space): ");
```

```
    scanf("%d %d", &m, &n);   // & is required for scanf
    m = m + 10;
    n = 5 * m;                // corrected from '5^ * m' to '5 * m'
    printf("m = %d \t\t\t n = %d\n", m, n);
}
```

Input:

10 15

Execution:

- **Initially**, m and n contain garbage values.
- After scanf("%d %d", &m, &n);, m = 10 and n = 15.
- m = m + 10; updates m to 20.
- n = 5 * m; updates n to 100.
- printf("m = %d \t\t\t n = %d\n", m, n); displays the values with tab spaces.

Output:

m = 20 n = 100

🌟 b) Contents of memory for variables 'm' and 'n' before and after execution

- Before scanf: m and n contain random (garbage) values.
- After scanf: m = 10, n = 15 (values entered by the user are stored).
- After m = m + 10: m = 20, n = 15.
- After n = 5 * m: m = 20, n = 100.
- Memory values change step by step as operations are performed.

🌟 c) Write the program using scanf() function

C

```
#include <stdio.h>

void main() {

   int m, n;

   // Take input from user

   printf("Enter two numbers (separated by space): ");

   scanf("%d %d", &m, &n);

   // Perform calculations

   m = m + 10;

   n = 5 * m;
```

```
    // Display output
    printf("m = %d \t\t\t n = %d\n", m, n);
}
```

Key Exam Points:

- Always use & with variables in scanf().
- Use * for multiplication, not ^.
- \t adds tab space, \n moves to next line.
- Know how memory values change during execution.

🌟 **Q.8:**

(a) Printing -17.246 using different formats

Concept:

- %m.nf ⟶ m = total field width, n = number of decimal places
- Rounding occurs automatically based on next decimal digit:
    - ≥5 ⟶ round up
    - <5 ⟶ round down

Output:

1. %8.4f ⟶ 8 spaces, 4 decimal places ⟶ -17.2460
2. %8.3f ⟶ 8 spaces, 3 decimal places ⟶ -17.246
3. %8.2f ⟶ 8 spaces, 2 decimal places ⟶ -17.25
4. %8.1f ⟶ 8 spaces, 1 decimal place ⟶ -17.2
5. %8.0f ⟶ 8 spaces, 0 decimal places ⟶ -17
6. %0.2f ⟶ no field width, 2 decimal places ⟶ -17.25

✅ **Note:** Field width ensures right-alignment, negative sign and decimal included in width.

🌟 **(b) Output of printf statements for x = 21.335 and y = 200**

Given:

- x = 21.335 (double)
- y = 200 (int)

Statements and Output:

1. printf("x is %6.2f \t y is %4d\n", x, y);

- %6.2f ⟶ round x to 2 decimals ⟶ 21.34, total 6 spaces
- %4d ⟶ right-align y in 4 spaces ⟶ 200
- \t ⟶ tab

Output:

x is 21.34 y is 200

2. printf("y is %d\n", y);

- %d ⟶ prints integer

Output:

y is 200

3. printf("x is %.1f\n", x);

- %.1f ⟶ round x to 1 decimal ⟶ 21.3

Output:

x is 21.3

✅ **Key points**: %m.nf ⟶ m = field width, n = precision; %d ⟶ integer.

🌟 **(c) Display variables a = 307, b = 408.558, c = -12.31 as 307 408.56-12.3**

Required Output:

307 408.56-12.3

Correct printf Statement:

C

printf("%d %.2f%.1f", a, b, c);

Explanation:

%d ⟶ prints a = 307

%.2f ⟶ prints b = 408.558 rounded to 2 decimals ⟶ 408.56

%.1f ⟶ prints c = -12.31 rounded to 1 decimal ⟶ -12.3

No space between b and c to match the required format.

✦ **Summary:**

- %m.nf ⟶ m = field width, n = decimal places
- %d ⟶ integer, %f ⟶ float/double
- Rounding occurs automatically
- \t ⟶ tab, \n ⟶ newline
- No space between variables if output requires concatenation

🌟 **Q.9:**

Write a program that asks the user to enter the radius of a circle and then computes and displays the circle's area. Use the formula:

area = PI * radius * radius

where PI is the constant value 3.14159 (use #define macro).

❖ **Answer (C Program)**

C

```
#include <stdio.h>

#define PI 3.14159  // Constant macro for PI

void main() {

   float radius, area;

   // Ask the user to enter radius

   printf("Enter the radius of the circle: ");

   scanf("%f", &radius);  // %f for float input, &radius stores input in variable

   // Calculate area using formula

   area = PI * radius * radius;  // Plain text: area = PI * radius * radius

   // Display the area
```

```
    printf("The area of the circle with radius %.2f is %.2f\n",
radius, area);

}
```

Explanation:

1. #define PI 3.14159 ⟶ defines constant PI using macro
2. float radius, area; ⟶ variables to store radius and area (float for decimal values)
3. scanf("%f", &radius); ⟶ takes input from user, & is used to store value at the variable's address
4. area = PI * radius * radius; ⟶ calculates area using plain text formula
5. printf("The area ... %.2f"); ⟶ displays the area, %.2f rounds the number to 2 decimal places

Sample Output

Enter the radius of the circle: 7

The area of the circle with radius 7.00 is 153.94

✅ Exam Tips:

- Always use macro for constants like PI.
- Use float or double for decimal calculations.

- & with scanf is mandatory to store input correctly.
- Round output using %.2f for clarity in exams.

🌟 **Q.10: Write a program that stores the values 'A', 'U', 3.456E10 and 50 in separate memory cells. Your program should get the first three values as input data, but use an assignment statement to store the last value.**

❖ **Answer (C Program)**

C

```
#include <stdio.h>

void main() {
    char ch1, ch2;
    double num;
    int val;
    // Input first three values from user
    printf("Enter first character: ");
    scanf(" %c", &ch1);  // note the space before %c
    printf("Enter second character: ");
```

```
    scanf(" %c", &ch2);  // note the space before %c
    printf("Enter a floating point number (e.g., 3.456E10): ");
    scanf("%lf", &num);  // %lf for double
    // Assign the last value using assignment statement
    val = 50;
    // Display all values
    printf("\nValues stored in memory cells are:\n");
    printf("Character 1: %c\n", ch1);
    printf("Character 2: %c\n", ch2);
    printf("Floating point number: %.2E\n", num); // scientific notation
    printf("Integer value: %d\n", val);
}
```

Explanation:

1. Variable Declaration:

- **char ch1, ch2;** → to store the two characters 'A' and 'U'
- **double num;** → to store 3.456E10 (scientific notation)

- int val; ⟶ to store integer 50

2. Input using scanf():

- **scanf(" %c", &ch1);** ⟶ space before %c ensures previous input does not interfere
- **scanf("%lf", &num);** ⟶ %lf is used for double type input

3. Assignment Statement:

val = 50; ⟶ stores integer 50 directly without user input

4. Output using printf():

- %c ⟶ prints character
- %.2E ⟶ prints floating point number in scientific notation with 2 decimal places
- %d ⟶ prints integer

Sample Run

Enter first character: A

Enter second character: U

Enter a floating point number (e.g., 3.456E10): 3.456E10

Values stored in memory cells are:

Character 1: A

Character 2: U

Floating point number: 3.46E10

Integer value: 50

✅ **Exam Tips:**

- **For characters**, always use %c and space before %c in scanf to avoid input issues
- For double, use %lf in scanf
- Direct assignment is used for values that do not require input from the user
- Scientific notation can be printed using %.2E or %.2e

🌟 **Q.11: Write a program that converts a temperature in degrees Fahrenheit to degrees Celsius.**

Formula:

celsius = 5/9 * (fahrenheit - 32)

❖ **Answer (C Program)**

C

#include <stdio.h>

void main() {

```
    float fahrenheit, celsius;

    // Ask the user to enter temperature in Fahrenheit

    printf("Enter temperature in Fahrenheit: ");

    scanf("%f", &fahrenheit);

    // Convert Fahrenheit to Celsius

    celsius = 5.0 / 9.0 * (fahrenheit - 32);  // Use 5.0/9.0 to get
decimal result

    // Display the result

    printf("Temperature in Celsius is: %.2f\n", celsius);

}
```

Explanation:

1. Variable Declaration:

- float fahrenheit, celsius; ⟶ use float for decimal values

2. Input using scanf():

- scanf("%f", &fahrenheit); ⟶ %f for float input, & stores the value in variable

3. Conversion Formula:

- celsius = 5.0 / 9.0 * (fahrenheit - 32); → Plain text version: celsius = 5/9 * (fahrenheit - 32)
- Use 5.0/9.0 instead of 5/9 to ensure decimal division, otherwise result will be zero in integer division

4. Output using printf():

%.2f ⟶ rounds the result to 2 decimal places

Sample Run

Enter temperature in Fahrenheit: 98.6

Temperature in Celsius is: 37.00

✅ **Exam Tips:**

- Always use decimal numbers in division for correct float result (5.0/9.0)
- Use float for variables to store decimal temperatures
- **Formula**: celsius = 5/9 * (fahrenheit - 32)

🌟 **Q.12: Write a program that takes a positive number with a fractional part and rounds it to two decimal places.**

Example:

- **Input**: 25.4851 ⟶ Output: 25.49

- **Input**: 62.4431 → Output: 62.44

❖ **Answer (C Program)**

C

```
#include <stdio.h>
#include <math.h>  // For round() function
void main() {
    double num, rounded;
    // Input a positive number
    printf("Enter a positive number: ");
    scanf("%lf", &num);  // %lf for double input
    // Round to two decimal places
    rounded = round(num * 100) / 100;  // Multiply by 100, round, then divide by 100
    // Display the rounded value
    printf("Number rounded to two decimal places: %.2f\n", rounded);
}
```

Explanation:

1. Variable Declaration:

- double num, rounded; ⟶ double for decimal numbers

2. Input using scanf():

- scanf("%lf", &num); ⟶ %lf used for double type input

3. Rounding Formula:

- Multiply the number by 100 ⟶ num * 100
- Apply the round() function ⟶ round(num * 100)
- Divide by 100 to get two decimal places ⟶ round(num * 100) / 100

4. Output using printf():

- %.2f ⟶ ensures exactly two decimal places are displayed

Sample Run

Enter a positive number: 25.4851

Number rounded to two decimal places: 25.49

Enter a positive number: 62.4431

Number rounded to two decimal places: 62.44

✅ Exam Tips:

- Always multiply by 10^n (here 100) before rounding to preserve the desired decimal places.
- round() returns nearest integer; divide by 100 to get decimal rounded value.
- Use %.2f in printf to display exactly 2 decimal places.

Note:

This chapter is designed to provide a solid foundation of knowledge, with the goal of deepening understanding and encouraging further exploration of the subject. The content has been carefully selected to support effective learning and inspire students to engage with the topic more deeply.

Author: Muhammad Asghar

Purpose: To contribute to education by offering insightful, valuable content that enhances learning and understanding.