

# Programming in C (4331105) - Summer 2025 Solution

Milav Dabgar

May 20, 2025

## Question 1 [a marks]

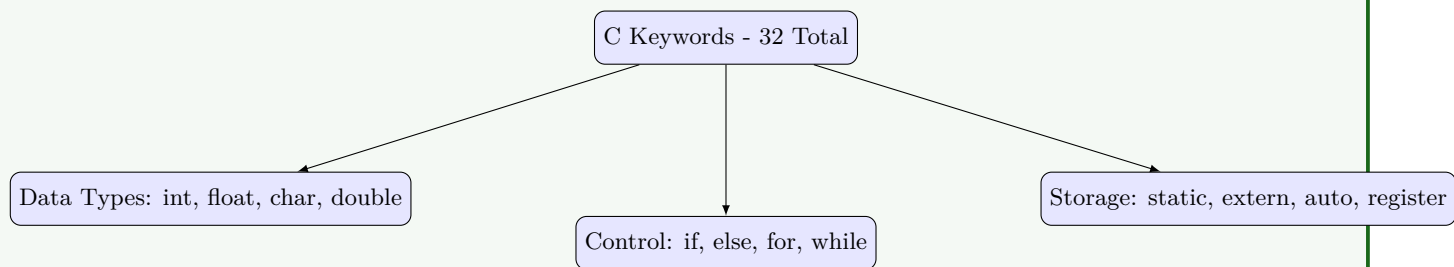
3 How many keywords are there in C? Write any four keywords

### Solution

Answer:

Total Keywords	Examples
32 keywords	int, float, char, if

Diagram:



- **32 keywords:** Total reserved words in C language
- **Data type keywords:** int, float, char, double for variable declaration
- **Control keywords:** if, else, for, while for program flow

### Mnemonic

"Cats In Four Colors" (char, int, float, const)

## Question 1 [b marks]

4 What is variable? Explain rules for naming a variable with example

### Solution

Answer:

**Variable Definition:**

Aspect	Description
Definition	Named memory location to store data
Purpose	Hold values that can change during program execution
Declaration	<code>datatype variable_name;</code>

**Naming Rules:**

- **First character:** Must be letter or underscore (\_)
- **Subsequent characters:** Letters, digits, underscore only
- **Case sensitive:** 'Age' and 'age' are different
- **No keywords:** Cannot use reserved words like 'int', 'float'

Examples:

```

1 int age;           // Valid
2 float _salary;    // Valid
3 char name123;     // Valid
4 int 2number;      // Invalid - starts with digit
5 float for;        // Invalid - keyword used

```

#### Mnemonic

"Letters First, No Keywords" (LF-NK)

## Question 1 [c marks]

7 Specify errors if any, in the following statements

#### Solution

Answer:

Statement	Error	Reason
(1) fLoat x;	Invalid keyword	Correct: float x;
(2) int min, max = 20;	Partial initialization	Only max initialized, min uninitialized
(3) long char c;	Invalid combination	Cannot combine long with char
(4) iNt a;	Invalid keyword	Correct: int a;
(5) FLOAT f=2;	Invalid keyword	Correct: float f=2;
(6) double m ; n;	Missing datatype	Correct: double m, n;
(7) Int score (100)0;	Multiple errors	Invalid syntax, correct: int score = 100;

Key Points:

- **Case sensitivity:** Keywords must be lowercase
- **Multiple declaration:** Use comma separator
- **Initialization syntax:** Use = operator

#### Mnemonic

"Keywords Lower Case Always" (KLCA)

OR

## Question 1 [c marks]

7 What is algorithm? What is flowchart? Draw a flowchart to find area and perimeter of circle.

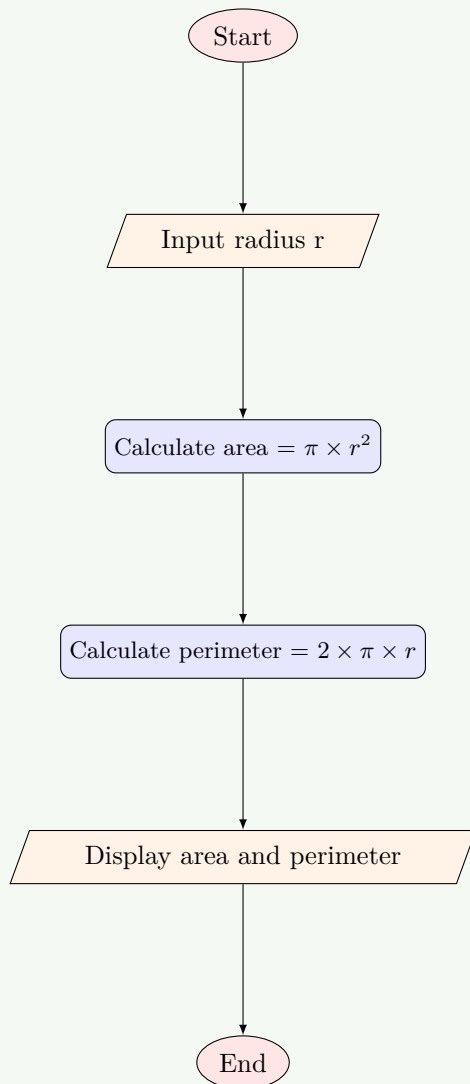
#### Solution

Answer:

Definitions:

Term	Definition
Algorithm	Step-by-step procedure to solve a problem
Flowchart	Visual representation of algorithm using symbols

**Flowchart for Circle Area and Perimeter:**



**Algorithm Steps:**

- **Step 1:** Start
- **Step 2:** Input radius value
- **Step 3:** Calculate area using formula  $\pi \times r^2$
- **Step 4:** Calculate perimeter using formula  $2 \times \pi \times r$

**Mnemonic**

"Start Input Calculate Display End" (SICDE)

## Question 2 [a marks]

**3** What is operator? List all the 'C' operators.

**Solution****Answer:****Operator Definition:**

Aspect	Description
Definition	Special symbols that perform operations on operands
Purpose	Manipulate data and variables

**C Operators List:**

Category	Operators
Arithmetic	+, -, *, /, %
Relational	<, >, <=, >=, ==, !=
Logical	&&,   , !
Assignment	=, +=, -=, *=, /=
Increment/Decrement	++, --
Conditional	?:

**Mnemonic**

"Add Relate Logic Assign Increment Condition" (ARLIC)

**Question 2 [b marks]****4 State difference between while and do while loop.****Solution****Answer:**

Aspect	while loop	do-while loop
Entry condition	Pre-tested	Post-tested
Minimum execution	0 times	At least 1 time
Syntax	while(condition) { }	do { } while(condition);
Semicolon	Not required after while	Required after while

**Example:**

```

1 // while loop
2 while(i < 5) {
3     printf("%d", i);
4     i++;
5 }
6
7 // do-while loop
8 do {
9     printf("%d", i);
10    i++;
11 } while(i < 5);

```

**Key Points:**

- **Pre-tested:** Condition checked before execution
- **Post-tested:** Condition checked after execution

**Mnemonic**

"While Before, Do After" (WB-DA)

## Question 2 [c marks]

7 How is scanf() function used for formatted input? Explain with example

### Solution

Answer:

**scanf() Function:**

Feature	Description
Purpose	Read formatted input from keyboard
Syntax	scanf("format_string", &variable);
Return	Number of successfully read inputs

**Format Specifiers:**

Specifier	Data Type
%d	int
%f	float
%c	char
%s	string

**Examples:**

```

1  int age;
2  float salary;
3  char grade;
4
5  scanf("%d", &age);           // Read integer
6  scanf("%f", &salary);       // Read float
7  scanf("%c", &grade);        // Read character
8  scanf("%d %f", &age, &salary); // Multiple inputs

```

**Important Points:**

- **Address operator (&):** Required for variables
- **Format string:** Must match data types
- **Buffer issues:** Use fflush(stdin) if needed

### Mnemonic

"Address Format Match" (AFM)

OR

## Question 2 [a marks]

3 List arithmetic and relational operators of C language

### Solution

Answer:

Operator Type	Operators	Purpose
Arithmetic	+, -, *, /, %	Mathematical operations
Relational	<, >, <=, >=, ==, !=	Comparison operations

**Examples:**

```

1  // Arithmetic
2  int a = 10 + 5;    // Addition
3  int b = 10 % 3;    // Modulus (remainder)

```

```
4  
5 // Relational  
6 if(a > b)           // Greater than  
7 if(a == b)          // Equal to
```

**Mnemonic**

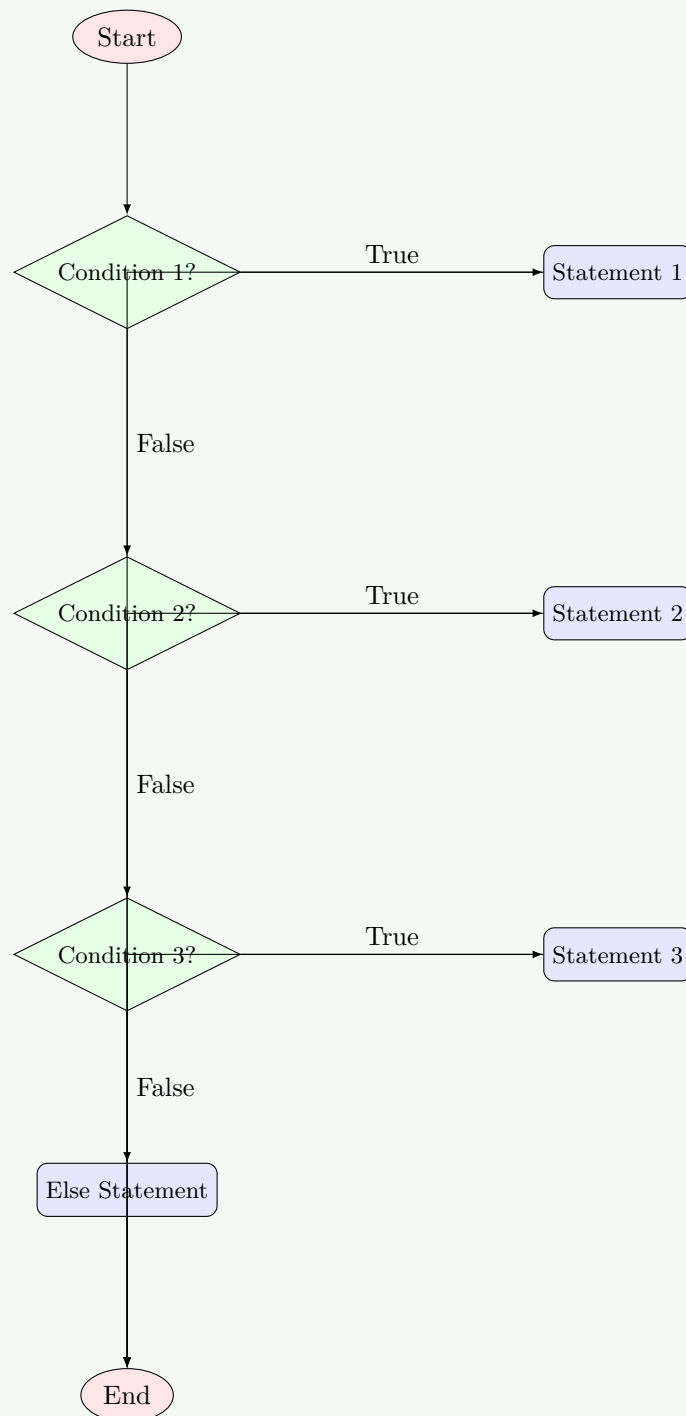
"Add Multiply Compare" (AMC)

## Question 2 [b marks]

4 Draw flow chart of else if ladder.

**Solution**

**Answer:**

**Structure:**

- **Multiple conditions:** Checked sequentially
- **First true:** Corresponding block executes
- **Default case:** Else block for no match

**Mnemonic**

"Check First True Execute" (CFTE)

## Question 2 [c marks]

7 How is printf() function used for formatted output? Explain with example

### Solution

Answer:

**printf() Function:**

Feature	Description
Purpose	Display formatted output on screen
Syntax	<code>printf("format_string", variables);</code>
Return	Number of characters printed

**Format Specifiers:**

Specifier	Usage	Example
%d	Integer	<code>printf("%d", 25);</code>
%f	Float	<code>printf("%.2f", 3.14);</code>
%c	Character	<code>printf("%c", 'A');</code>
%s	String	<code>printf("%s", "Hello");</code>

**Advanced Formatting:**

```

1  int num = 123;
2  float pi = 3.14159;
3
4  printf("Number: %5d\n", num);      // Width specification
5  printf("Pi: %.2f\n", pi);          // Precision specification
6  printf("Hex: %x\n", num);          // Hexadecimal
7  printf("Left aligned: %-10d\n", num); // Left alignment

```

**Escape Sequences:**

- \n: New line
- \t: Tab space
- \\: Backslash

### Mnemonic

"Format Width Precision Align" (FWPA)

## Question 3 [a marks]

3 List Logical operators and explain it

### Solution

Answer:

Operator	Symbol	Description	Truth Table
AND	&&	True if both operands true	T&&T = T, others = F
OR		True if any operand true	F  F = F, others = T
NOT	!	Inverts the condition	!T = F, !F = T

**Examples:**

```

1  int a = 5, b = 10;
2
3  if(a > 0 && b > 0)    // Both conditions must be true
4  if(a > 15 || b > 5)    // At least one condition true
5  if(!(a > 10))          // Negation of condition

```

**Mnemonic**

"And Or Not" (AON)

**Question 3 [b marks]**

4 Explain for loop with example.

**Solution****Answer:****For Loop Structure:**

Component	Purpose
Initialization	Set starting value
Condition	Test for continuation
Update	Modify loop variable

**Syntax:**

```

1 for(initialization; condition; update) {
2     statements;
3 }
```

**Example:**

```

1 // Print numbers 1 to 5
2 for(int i = 1; i <= 5; i++) {
3     printf("%d ", i);
4 }
5 // Output: 1 2 3 4 5
```

**Execution Flow:**

- **Step 1:** Initialize  $i = 1$
- **Step 2:** Check condition  $i \leq 5$
- **Step 3:** Execute statements
- **Step 4:** Update  $i++$ , repeat from step 2

**Mnemonic**

"Initialize Check Execute Update" (ICEU)

**Question 3 [c marks]**

7 Write a program to find maximum out of three integer numbers x and y.

**Solution****Answer:**

```

1 #include <stdio.h>
2
3 int main() {
4     int x, y, z, max;
```

```

5
6     printf("Enter three numbers: ");
7     scanf("%d %d %d", &x, &y, &z);
8
9     max = x; // Assume first number is maximum
10
11     if(y > max) {
12         max = y;
13     }
14     if(z > max) {
15         max = z;
16     }
17
18     printf("Maximum number is: %d", max);
19
20     return 0;
21 }

```

**Algorithm Steps:**

Step	Action
1	Input three numbers
2	Assume first as maximum
3	Compare with second, update if larger
4	Compare with third, update if larger
5	Display maximum

**Alternative Method:**

```

1 max = (x > y) ? ((x > z) ? x : z) : ((y > z) ? y : z);

```

**Mnemonic**

"Assume Compare Update Display" (ACUD)

OR

## Question 3 [a marks]

3 Explain conditional operator with example.

**Solution**

**Answer:**

**Conditional Operator (Ternary):**

Feature	Description
Symbol	?:
Syntax	condition ? value1 : value2
Purpose	Shortcut for if-else

**Examples:**

```

1 int a = 10, b = 20;
2 int max = (a > b) ? a : b;           // max = 20
3
4 char grade = (marks >= 60) ? 'P' : 'F';
5 printf("Status: %s", (age >= 18) ? "Adult" : "Minor");

```

**Equivalent if-else:**

```

1  if(a > b)
2      max = a;
3  else
4      max = b;

```

**Advantages:**

- **Concise:** Single line expression
- **Efficient:** Faster execution

**Mnemonic**

"Question Mark Colon Choice" (QMCC)

## Question 3 [b marks]

4 Explain while loop with example.

**Solution**

**Answer:**

**While Loop:**

Feature	Description
Type	Entry-controlled loop
Syntax	<code>while(condition) { statements; }</code>
Execution	Repeats while condition is true

**Example:**

```

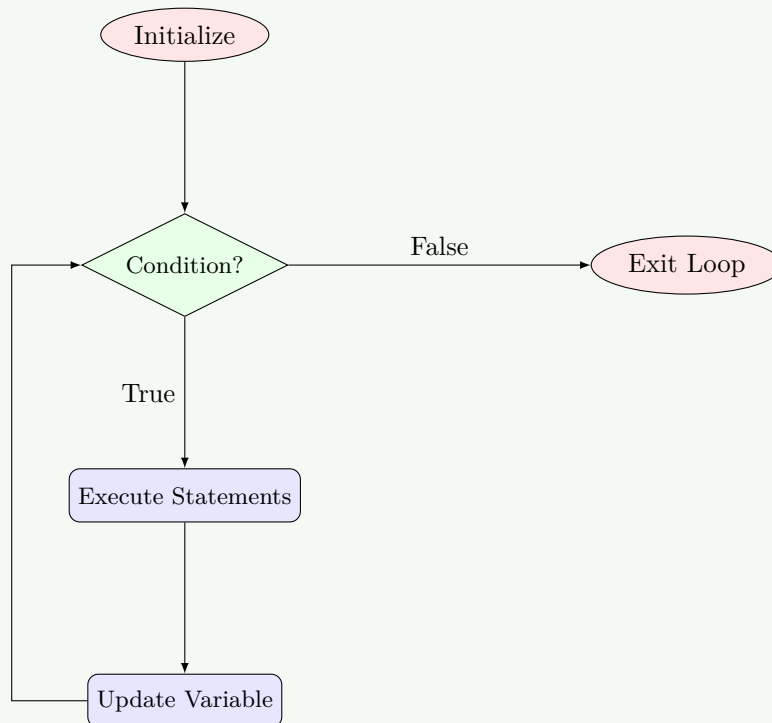
1  int i = 1;
2  while(i <= 5) {
3      printf("%d ", i);
4      i++;
5  }
6  // Output: 1 2 3 4 5

```

**Important Points:**

- **Initialization:** Before loop
- **Condition:** Checked at beginning
- **Update:** Inside loop body
- **Infinite loop:** If condition never becomes false

**Flowchart Structure:**

**Mnemonic**

"Initialize Check Execute Update" (ICEU)

## Question 3 [c marks]

7 WAP to read an integer from key board and print whether given number is odd or even.

### Solution

**Answer:**

```
1  #include <stdio.h>
2
3  int main() {
4      int number;
5
6      printf("Enter an integer: ");
7      scanf("%d", &number);
8
9      if(number % 2 == 0) {
10         printf("%d is Even number", number);
11     }
12     else {
13         printf("%d is Odd number", number);
14     }
15
16     return 0;
17 }
```

**Logic Explanation:**

Concept	Description
<b>Modulus operator (%)</b>	Returns remainder after division
<b>Even condition</b>	<code>number % 2 == 0</code>
<b>Odd condition</b>	<code>number % 2 != 0</code>

Alternative Methods:

```

1 // Method 2: Using conditional operator
2 printf("%d is %s", number, (number % 2 == 0) ? "Even" : "Odd");
3
4 // Method 3: Using bitwise AND
5 if(number & 1)
6     printf("Odd");
7 else
8     printf("Even");

```

Sample Output:

```

1 Enter an integer: 7
2 7 is Odd number

```

#### Mnemonic

"Modulus Two Zero Even" (MTZE)

## Question 4 [a marks]

3 Evaluate following arithmetic expressions:  $30/4*4 - 20\%6 + 17/2$

### Solution

Answer:

Step-by-step Evaluation:

Step	Expression	Calculation	Result
1	$30/4*4$	$(30/4)*4 = 7*4$	28
2	$20\%6$	$20 \bmod 6$	2
3	$17/2$	Integer division	8
4	Final	$28 - 2 + 8$	34

Operator Precedence:

Priority	Operators
High	$*, /, \%$ (Left to right)
Low	$+, -$ (Left to right)

Complete Calculation:

```

1 30/4*4 - 20%6 + 17/2
2 = 7*4 - 2 + 8      // Division and modulus first
3 = 28 - 2 + 8       // Multiplication
4 = 26 + 8           // Left to right for +,-
5 = 34               // Final answer

```

#### Mnemonic

"Multiply Divide Before Add Subtract" (MDBAS)

## Question 4 [b marks]

4 WAP to find sum and average of an array of 5 integer numbers.

### Solution

Answer:

```

1  #include <stdio.h>
2
3  int main() {
4      int numbers[5];
5      int sum = 0;
6      float average;
7
8      printf("Enter 5 integers:\n");
9      for(int i = 0; i < 5; i++) {
10         scanf("%d", &numbers[i]);
11         sum += numbers[i];
12     }
13
14     average = (float)sum / 5;
15
16     printf("Sum = %d\n", sum);
17     printf("Average = %.2f", average);
18
19     return 0;
20 }
```

**Algorithm:**

1. Declare array of 5 integers
2. Initialize sum to 0
3. Input 5 numbers using loop
4. Add each number to sum
5. Calculate average = sum/5
6. Display results

**Key Points:**

- **Type casting:** (float)sum for accurate division
- **Loop usage:** Efficient for repetitive input

### Mnemonic

"Declare Input Add Calculate Display" (DIACD)

## Question 4 [c marks]

7 Define pointer. Explain how pointers are declared and initialized with example.

### Solution

Answer:

**Pointer Definition:**

Aspect	Description
Definition	Variable that stores memory address of another variable
Purpose	Direct memory access and dynamic memory allocation
Symbol	* (asterisk) for declaration and dereferencing

**Declaration and Initialization:**

```

1 // Declaration
2 int *ptr;           // Pointer to integer
3 float *fptr;        // Pointer to float
4 char *cptr;         // Pointer to character
5
6 // Initialization
7 int num = 10;
8 int *ptr = &num;    // Initialize with address of num
9
10 // Alternative
11 int *ptr;
12 ptr = &num;         // Assign address later

```

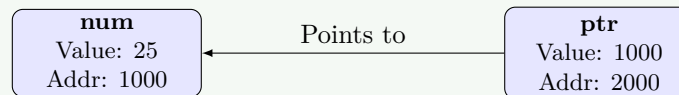
#### Example Program:

```

1 #include <stdio.h>
2
3 int main() {
4     int num = 25;
5     int *ptr = &num;
6
7     printf("Value of num: %d\n", num);
8     printf("Address of num: %p\n", &num);
9     printf("Value of ptr: %p\n", ptr);
10    printf("Value pointed by ptr: %d\n", *ptr);
11
12    return 0;
13 }

```

#### Memory Diagram:



#### Mnemonic

"Address Star Dereference" (ASD)

OR

## Question 4 [a marks]

3 Evaluate following arithmetic expressions:  $50 / 3 \% 3 + 5 * 7$

#### Solution

Answer:

Step-by-step Evaluation:

Step	Expression	Calculation	Result
1	50/3	Integer division	16
2	16%3	16 mod 3	1
3	5*7	Multiplication	35
4	Final	1 + 35	36

Complete Calculation:

```

1 50 / 3 % 3 + 5 * 7

```

```
2 = 16 % 3 + 35    // Division and multiplication first
3 = 1 + 35         // Modulus operation
4 = 36             // Final answer
```

**Operator Precedence Applied:**

- **High priority:** /, %, \* (left to right)
- **Low priority:** + (left to right)

**Mnemonic**

"Divide Mod Multiply Add" (DMMA)

## Question 4 [b marks]

4 WAP to find the largest number in an array of N integers.

**Solution****Answer:**

```
1 #include <stdio.h>
2
3 int main() {
4     int n, i;
5     int largest;
6
7     printf("Enter number of elements: ");
8     scanf("%d", &n);
9
10    int arr[n];
11
12    printf("Enter %d numbers:\n", n);
13    for(i = 0; i < n; i++) {
14        scanf("%d", &arr[i]);
15    }
16
17    largest = arr[0]; // Assume first element is largest
18
19    for(i = 1; i < n; i++) {
20        if(arr[i] > largest) {
21            largest = arr[i];
22        }
23    }
24
25    printf("Largest number is: %d", largest);
26
27    return 0;
28 }
```

**Algorithm:**

1. Input array size
2. Input array elements
3. Assume first element as largest
4. Compare with remaining elements
5. Update largest if bigger found
6. Display result

**Mnemonic**

"Input Assume Compare Update Display" (IACUD)

**Question 4 [c marks]**

7 Define array. Explain the need for array variable. Explain 1-D array with example

**Solution**

**Answer:**

**Array Definition:**

Aspect	Description
Definition	Collection of similar data type elements
Storage	Consecutive memory locations
Access	Using index/subscript

**Need for Arrays:**

Problem	Solution with Array
Store multiple values	Single array variable
Avoid multiple variables	arr[100] instead of a1, a2, ..., a100
Efficient processing	Loop-based operations
Memory organization	Contiguous allocation

**1-D Array Declaration:**

```

1 datatype arrayname[size];
2
3 // Examples
4 int marks[5];           // Array of 5 integers
5 float prices[10];       // Array of 10 floats
6 char name[20];          // Array of 20 characters

```

**Array Initialization:**

```

1 // Method 1: At declaration
2 int numbers[5] = {10, 20, 30, 40, 50};
3
4 // Method 2: Individual assignment
5 int arr[3];
6 arr[0] = 5;
7 arr[1] = 15;
8 arr[2] = 25;

```

**Example Program:**

```

1 #include <stdio.h>
2
3 int main() {
4     int marks[5] = {85, 90, 78, 92, 88};
5     int i, sum = 0;
6
7     printf("Student marks:\n");
8     for(i = 0; i < 5; i++) {
9         printf("Subject %d: %d\n", i+1, marks[i]);
10        sum += marks[i];
11    }
12

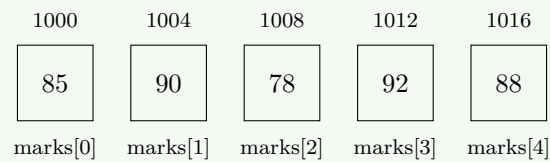
```

```

13     printf("Total marks: %d", sum);
14     return 0;
15 }

```

#### Memory Layout:



#### Mnemonic

"Similar Data Consecutive Index" (SDCI)

## Question 5 [a marks]

3 Give an example of if ... else statement.

### Solution

#### Answer:

```

1  #include <stdio.h>
2
3  int main() {
4      int age;
5
6      printf("Enter your age: ");
7      scanf("%d", &age);
8
9      if(age >= 18) {
10         printf("You are eligible to vote");
11     }
12     else {
13         printf("You are not eligible to vote");
14     }
15
16     return 0;
17 }

```

#### Structure:

Component	Purpose
if	Tests condition
condition	Boolean expression
if-block	Executes when condition true
else-block	Executes when condition false

#### Sample Outputs:

```

1  Input: 20    Output: You are eligible to vote
2  Input: 16    Output: You are not eligible to vote

```

#### Mnemonic

"If True Else False" (ITEF)

## Question 5 [b marks]

4 WAP to check the category of given character.

### Solution

Answer:

```

1  #include <stdio.h>
2  #include <ctype.h>
3
4  int main() {
5      char ch;
6
7      printf("Enter a character: ");
8      scanf("%c", &ch);
9
10     if(isdigit(ch)) {
11         printf("%c' is a Digit", ch);
12     }
13     else if(isupper(ch)) {
14         printf("%c' is an Uppercase letter", ch);
15     }
16     else if(islower(ch)) {
17         printf("%c' is a Lowercase letter", ch);
18     }
19     else {
20         printf("%c' is a Special symbol", ch);
21     }
22
23     return 0;
24 }
```

Character Categories:

Function	Category	Range
isdigit()	Digit	0-9
isupper()	Uppercase	A-Z
islower()	Lowercase	a-z
Others	Special symbols	!@#\$\$%^&* etc.

Alternative Method:

```

1  if(ch >= '0' && ch <= '9')
2      printf("Digit");
3  else if(ch >= 'A' && ch <= 'Z')
4      printf("Uppercase");
5  else if(ch >= 'a' && ch <= 'z')
6      printf("Lowercase");
7  else
8      printf("Special symbol");
```

### Mnemonic

"Digit Upper Lower Special" (DULS)

## Question 5 [c marks]

7 What is structure? Explain its syntax with suitable example

## Solution

**Answer:**

**Structure Definition:**

Aspect	Description
Definition	User-defined data type combining different data types
Purpose	Group related data under single name
Keyword	struct

**Syntax:**

```

1 struct structure_name {
2     datatype member1;
3     datatype member2;
4     ...
5 };

```

**Example - Student Structure:**

```

1 #include <stdio.h>
2
3 struct Student {
4     int roll_no;
5     char name[50];
6     float marks;
7     char grade;
8 };
9
10 int main() {
11     struct Student s1;
12
13     // Input data
14     printf("Enter roll number: ");
15     scanf("%d", &s1.roll_no);
16
17     printf("Enter name: ");
18     scanf("%s", s1.name);
19
20     printf("Enter marks: ");
21     scanf("%f", &s1.marks);
22
23     printf("Enter grade: ");
24     scanf(" %c", &s1.grade);
25
26     // Display data
27     printf("\nStudent Details:\n");
28     printf("Roll No: %d\n", s1.roll_no);
29     printf("Name: %s\n", s1.name);
30     printf("Marks: %.2f\n", s1.marks);
31     printf("Grade: %c\n", s1.grade);
32
33     return 0;
34 }

```

**Structure Features:**

- **Dot operator (.)**: Access structure members
- **Memory allocation**: Total size = sum of all members
- **Initialization**: Can initialize at declaration

**Structure Initialization:**

```

1 struct Student s1 = {101, "John", 85.5, 'A'};

```

**Memory Layout:**

**s1 Structure**

roll\_no (4 bytes)  
 name (50 bytes)  
 marks (4 bytes)  
 grade (1 byte)

**Mnemonic**

"Group Related Data Together" (GRDT)

OR

## Question 5 [a marks]

3 WAP to Print all numbers between -5 & +5.

### Solution

#### Answer:

```

1  #include <stdio.h>
2
3  int main() {
4      int i;
5
6      printf("Numbers between -5 and +5:\n");
7
8      for(i = -5; i <= 5; i++) {
9          printf("%d ", i);
10     }
11
12     return 0;
13 }
```

#### Output:

```

1  Numbers between -5 and +5:
2  -5 -4 -3 -2 -1 0 1 2 3 4 5
```

#### Alternative Methods:

```

1  // Method 2: Using while loop
2  int i = -5;
3  while(i <= 5) {
4      printf("%d ", i);
5      i++;
6  }
7
8  // Method 3: Two separate loops
9  for(i = -5; i < 0; i++)
10     printf("%d ", i);
11  printf("0 ");
12  for(i = 1; i <= 5; i++)
13     printf("%d ", i);
```

**Mnemonic**

"Start Negative End Positive" (SNEP)

**Question 5 [b marks]**

4 WAP to find roots of quadratic equation.

**Solution**

Answer:

```

1  #include <stdio.h>
2  #include <math.h>
3
4  int main() {
5      float a, b, c;
6      float discriminant, root1, root2;
7
8      printf("Enter coefficients (a, b, c): ");
9      scanf("%f %f %f", &a, &b, &c);
10
11     discriminant = b*b - 4*a*c;
12
13     if(discriminant > 0) {
14         root1 = (-b + sqrt(discriminant)) / (2*a);
15         root2 = (-b - sqrt(discriminant)) / (2*a);
16         printf("Roots are real and different\n");
17         printf("Root1 = %.2f\n", root1);
18         printf("Root2 = %.2f\n", root2);
19     }
20     else if(discriminant == 0) {
21         root1 = -b / (2*a);
22         printf("Roots are real and equal\n");
23         printf("Root = %.2f\n", root1);
24     }
25     else {
26         float realPart = -b / (2*a);
27         float imagPart = sqrt(-discriminant) / (2*a);
28         printf("Roots are complex\n");
29         printf("Root1 = %.2f + %.2fi\n", realPart, imagPart);
30         printf("Root2 = %.2f - %.2fi\n", realPart, imagPart);
31     }
32
33     return 0;
34 }
```

**Quadratic Formula Analysis:**

Discriminant	Nature of Roots
$b^2 - 4ac > 0$	Real and different
$b^2 - 4ac = 0$	Real and equal
$b^2 - 4ac < 0$	Complex (imaginary)

Formula:  $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

**Mnemonic**

"Discriminant Decides Root Nature" (DDRN)

## Question 5 [c marks]

7 Explain following built-in functions with examples

### Solution

Answer:

Function Explanations:

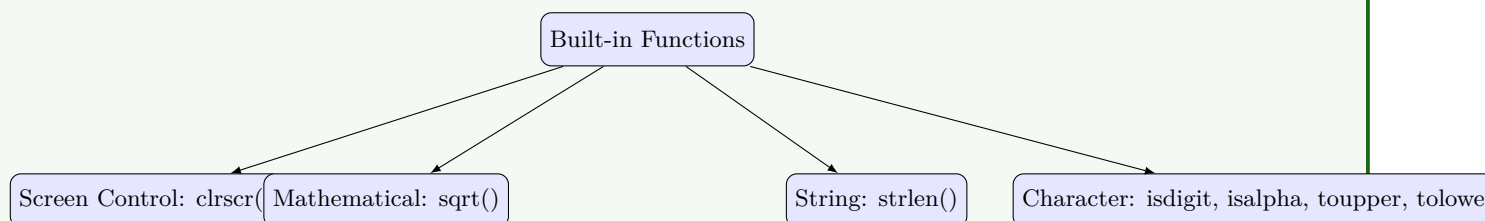
Function	Purpose	Header File	Example
clrscr()	Clear screen	conio.h	clrscr();
sqrt()	Square root	math.h	sqrt(16) = 4.0
strlen()	String length	string.h	strlen("Hello") = 5
isdigit()	Check if digit	ctype.h	isdigit('5') = true
isalpha()	Check if alphabet	ctype.h	isalpha('A') = true
toupper()	Convert to uppercase	ctype.h	toupper('a') = 'A'
tolower()	Convert to lowercase	ctype.h	tolower('B') = 'b'

Example Program:

```

1  #include <stdio.h>
2  #include <math.h>
3  #include <string.h>
4  #include <ctype.h>
5
6  int main() {
7      // clrscr(); // Not standard in modern compilers
8
9      // sqrt() example
10     float num = 25.0;
11     printf("Square root of %.1f = %.2f\n", num, sqrt(num));
12
13     // strlen() example
14     char str[] = "Programming";
15     printf("Length of '%s' = %d\n", str, strlen(str));
16
17     // Character functions
18     char ch = 'a';
19     printf("'%c' is digit: %s\n", ch, isdigit(ch) ? "Yes" : "No");
20     printf("'%c' is alphabet: %s\n", ch, isalpha(ch) ? "Yes" : "No");
21     printf("Uppercase of '%c' = '%c'\n", ch, toupper(ch));
22
23     return 0;
24 }
```

Function Categories:



### Mnemonic

"Clear Math String Character" (CMSC)