Unit-2 (C-Programming)

Topic: Decision Control Structures, Iterative Statements, Conditional Statements, Storage Classes, Arrays

1. Decision Control Structures

(a) If Statement

Executes a block of code only if the given condition is true.

```
if (condition) {
}
```

Example 1:

```
#include <stdio.h>

int main() {

int num = 10;

if (num > 0) {

printf("The number is positive.\n");

}

return 0;

}
```

Example 2:

(b) If-Else Statement

Executes one block if condition is true, another if false.

Example:

```
if (marks >= 50)
   printf("Pass");
else
   printf("Fail");
```

(c) Nested if Statement

An if inside another if.

Example:

```
if (a != b) {
   if (a < b)
      printf("a is smaller");
}
```

(d) If-Else Ladder

Multiple conditions checked one after another.

Example:

```
if (marks >= 90)
   printf("Grade A");
else if (marks >= 75)
   printf("Grade B");
else
   printf("Fail");
```

(e) Switch-Case Statement

Used when you need to choose one option from many.

Example:

```
#include <stdio.h>

int main() {

   char day;

   printf("Enter day (A-D): ");

   scanf(" %c", &day); // space before %c ignores any leftover newline


   switch (day) {

      case 'a':

         printf("Monday\n");

         break:

      case 'b':

         printf("Tuesday\n");

         break;
```

```
        case 'c':
            printf("Wednesday\n");
            break;
        case 'd':
            printf("Thursday\n");
            break;
        default:
            printf("Invalid day\n");
    }
    return 0;
}
```

2. Iterative Statements (Loops)

(a) For Loop

```
for (initialization; condition; update) {

}
```

Example:

```
for (int i = 1; i <= 5; i++) {
    printf("%d", i);
}
```

(b) While Loop

```
while (condition) {

}
```

Example:

```
int i=1;
while(i<=5){
   printf("%d",i);
   i++;
}
```

(c) Do-While Loop

```
do {


} while (condition);
```

Example:

```
int i=1;
do {
  printf("%d",i);
  i++;
} while(i<=5);
```

3. Conditional Statements

(1) Break → exits loop/switch immediately.

```
#include <stdio.h>

int main() {

for (int i = 1; i <= 10; i++) {

if (i == 5) {

break;

}

printf("%d ", i);

}

return 0;

}
```

Output: 1 2 3 4

(2) Continue → skips current iteration.

```
#include <stdio.h>
int main() {
for (int i = 1; i <= 5; i++) {
if (i == 3) {
continue;
}
printf("%d ", i);
}
return 0;
}
```

Output: 1 2 4 5

4. Storage Classes in C

Storage Class	Lifetime	Scope	Where Stored	Special Feature
`auto`	Function duration	Local	RAM	Default for local variables
`register`	Function duration	Local	CPU register (if possible)	Faster access, no `&` allowed
`static`	Entire program	Local/Global	RAM	Remembers value between calls
`extern`	Entire program	Global	RAM	Uses variable from another file/function

a) Auto

```
#include <stdio.h>
int main() {
   auto int a = 10;
   printf("%d\n", a);
   return 0;
}
```

b) Register

```
#include <stdio.h>
int main() {
   register int r = 20;
   printf("%d\n", r);
   return 0;
}
```

c) Static

```
#include <stdio.h>
int salman(int a, int b) {
  static int j;
   j++;
   int sum = j + (a + b);
   return sum;
}
int main() {
   int a, b, result;
   printf("Enter two numbers: ");
   scanf("%d %d", &a, &b);

   result = salman(a, b);
   printf("Sum after 1st call: %d\n", result);
   result = salman(a, b);
   printf("Sum after 2nd call: %d\n", result);
   result = salman(1, 1);
   printf("Sum after 3rd call: %d\n", result);
   return 0;
}
```

d) Extern

```
include <stdio.h>
int j=1;
int salman(int a, int b) {
```

```
    extern int j;
    j++;
    int sum = j + (a + b);
    return sum;
}
int main() {
    int a, b, sum, result;
 printf("Enter two numbers: ");
    scanf("%d \n %d", &a, &b);
  sum = j + (a+b);
    printf("Sum after 1st call: %d\n", sum);
    result = salman(a, b);
    printf("Sum after 2nd call: %d\n", result);
    result = salman(a, b);
    printf("Sum after 3rd call: %d\n", result);
    return 0;
}
```

5. Arrays

Definition: An array is a data structure that stores a fixed-size, sequential collection of elements of the same data type under a single variable name. Each element is accessed using an index, starting from 0.

Example: An array of integers [10, 20, 30] stores three numbers, accessible as array[0] = 10, array[1] = 20, and array[2] = 30.

Declaration of an Array: Array declaration specifies the data type, the array's name, and its size (number of elements). It reserves memory for the array but doesn't assign values yet.

Syntax :

data_type array_name[size];

Initialization of an Array: Initialization assigns values to the array elements at the time of declaration or later. You can initialize all elements at once or set specific values.

data_type array_name[size] = {value1, value2, ..., valueN};

1) **Full Initialization**:
int numbers[5] = {10, 20, 30, 40, 50};

2) **Partial Initialization**:

int numbers[5] = {1, 2}; // numbers[0] = 1, numbers[1] = 2, numbers[2] to numbers[4] = 0

3) **Initialization without Size**:

int numbers[] = {1, 2, 3}; // Size is inferred as 3

4) Later Initialization (using a loop):

```
int numbers[5];

for (int i = 0; i < 5; i++) {

   numbers[i] = i * 10; // Assigns 0, 10, 20, 30, 40

}
```

Types of Arrays : An Arrays can be classified based on their dimensions. In C language there is no limits on the number of dimensions in an Array.

a) **1 D Array (1 Dimensional Array):** A single-dimensional (1D) array is a linear list of elements of the same type, accessed using one index.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
85	90	78	92	56

int marks[5] = {85, 90, 78, 92,56};

```
#include <stdio.h>
int main() {
   int marks[5] = {85, 90, 78, 92,56};
   for (int i = 0; i < 4; i++) {
      printf("marks[%d] = %d\n", i, marks[i]);
   }
   return 0;
}
```

b) **2 D Array (Two-dimensional Array):**

A two-dimensional (2D) array is an array of arrays, often visualized as a table with rows and columns. It requires two indices: one for the row and one for the column.

Syntax:

data_type array_name[rows][columns];

Example:

int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};

2 rows and 3 columns

(0,0) 1	(0,1) 2	(0,2) 3
(1,0) 4	(1,1) 5	(1,2) 6

int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {40,14,90}};

3 rows and 3 columns

(0,0) 1	(0,1) 2	(0,2) 3
(1,0) 4	(1,2) 5	(1,2) 6
(2,0) 40	(2,1) 14	(2,2) 90

Exmp:

```
#include <stdio.h>

int main() {

   int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};

   for (int i = 0; i < 2; i++) {

      for (int j = 0; j < 3; j++) {

         printf("%d ", matrix[i][j]);

      }

      printf("\n");

   }

   return 0;

}
```

c) 3D Array (Three-Dimensional Array):

A **3D array** (three-dimensional array) is a data structure that stores elements in a three-dimensional grid, often visualized as a cube or a collection of 2D arrays stacked together. It uses three indices to access elements: one for the depth (or block), one for the row, and one for the column. This makes it suitable for representing data with three dimensions, such as 3D coordinates, volumetric data, or layered matrices.

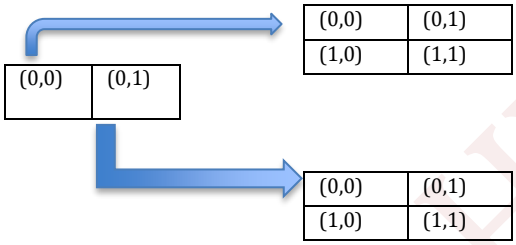
- **Structure**: A 3D array can be thought of as an array of 2D arrays, where each 2D array is a matrix, and the third dimension specifies which matrix (or layer) you're accessing.

Syntax:

data_type array_name[depth][rows][columns];

example:

int A [2] [2] [2];

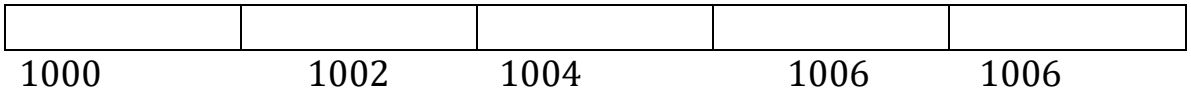


Example:

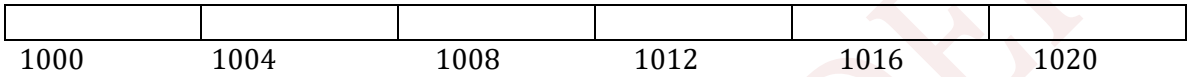
```
#include <stdio.h>

int main() {

   int array3D[2][2][3] = {

      {{1, 2, 3}, {4, 5, 6}},

      {{7, 8, 9}, {10, 11, 12}}

   };

   printf("Element at array3D[0][0][0] = %d\n", array3D[0][0][0]);

   printf("Element at array3D[0][1][2] = %d\n", array3D[0][1][2]);

   printf("Element at array3D[1][0][1] = %d\n", array3D[1][0][1]);

   printf("Element at array3D[1][1][2] = %d\n", array3D[1][1][2]);

   return 0;

}
```

Address calculation of an Array:

if element size = 2 bytes, and base address=1000 then memory address will be:



if element size = 4 bytes, and base address=1000 then memory address will be:

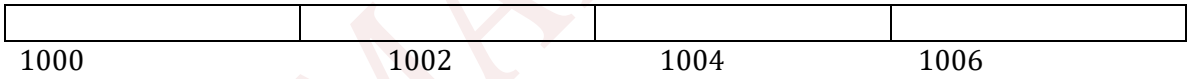


In A [2] [20]

if element size = 2 bytes, and base address=1000 then memory address will be:

1000	1002
1004	1006

OR we can simplify as



Formula for Calculate address:

- 1D: `Base + (i) * size`
- 2D: `Base + ((i * cols) + j) * size`
- 3D: `Base + ((i * rows * cols) + (j * cols) + k) * size`
- 4D: `Base + ((i*dim2*dim3*dim4) + (j*dim3*dim4) + (k*dim4) + l) * size`

Note: if A[3…..9] [6…12] the we calculate number of rows and column = **Upper limit-lower limit + 1**

e.g. 9-3+1=7, 12-6+1= 7

50 MCQs with Answers

A. Decision Control Structures

1. Which keyword is used for decision making in C?
   a) select
   b) if
   c) switch
   d) case
   Ans: b

2. Which statement is used to check multiple conditions in order?
   a) switch
   b) if-else ladder
   c) nested for
   d) continue
   Ans: b

3. Which statement executes one block if condition is true, and another if false?
   a) if
   b) if-else
   c) nested if
   d) switch
   Ans: b

4. In a switch statement, which keyword is optional?
   a) break
   b) case
   c) default
   d) both a and c
   Ans: c

5. Switch case cannot work with:
   a) int
   b) char
   c) float
   d) enum
   Ans: c

6. Which statement supports multiple options selection?
   a) switch-case
   b) nested if
   c) if-else ladder
   d) continue
   Ans: a

7. What will be the output?

```
int a=5;
```

if(a==5) printf("Yes");

else printf("No");

a) Yes
b) No
c) Error
d) Nothing
Ans: a

8. What happens if break is missing in a switch case?
   a) Compile-time error
   b) Execution stops
   c) Fall-through to next case
   d) None
   Ans: c

B. Loops

9. Which loop is entry-controlled?
   a) for
   b) while
   c) do-while
   d) both a & b
   Ans: d

10. Which loop executes at least once?
    a) for
    b) while
    c) do-while
    d) none
    Ans: c

11. An infinite loop can be created using:
    a) while(1)
    b) for(;;)
    c) do{ }while(1)
    d) all of the above
    Ans: d

12. Which loop is best when number of iterations is known?
    a) while
    b) for
    c) do-while
    d) switch
    Ans: b

13. Which loop is best for menu-driven programs?
    a) for
    b) while
    c) do-while
    d) nested for
    Ans: c

14. Which part of for loop is optional?
    a) initialization
    b) condition
    c) increment/decrement
    d) all of the above
    Ans: d

15. Output of:

```
for(int i=1; i<=3; i++){

  if(i==2) continue;

  printf("%d", i);

}
```

a) 1 2 3
b) 1 3
c) 2 3
d) Error
Ans: b

C. Conditional Statements (Break & Continue)

16. Break statement is used for:
    a) Exiting loop
    b) Exiting switch
    c) Both a & b
    d) None
    Ans: c

17. Continue statement is used for:
    a) Ending loop permanently
    b) Skipping current iteration
    c) Exiting program
    d) None
    Ans: b

18. Which transfers control to the beginning of the loop immediately?
    a) break
    b) continue

c) goto
d) return
Ans: b

19. Which is used to exit function?
    a) break
    b) exit()
    c) return
    d) continue
    Ans: c

20. Output of:

```
for(int i=1;i<=5;i++){

  if(i==3) break;

  printf("%d ", i);

}
```

a) 1 2 3 4 5
b) 1 2
c) 1 2 3
d) 3 4 5
Ans: b

D. Storage Classes

21. Default storage class of local variable?
    a) auto
    b) static
    c) extern
    d) register
    Ans: a

22. Which storage class preserves value between function calls?
    a) auto
    b) static
    c) extern
    d) register
    Ans: b

23. Which storage class stores variables in CPU registers?
    a) auto
    b) static
    c) extern
    d) register
    Ans: d

24. Global variables are by default:
    a) auto
    b) extern
    c) static
    d) register
    Ans: b

25. Default value of static variable?
    a) garbage
    b) 1
    c) 0
    d) undefined
    Ans: c

E. Arrays

26. Array index starts from?
    a) 0
    b) 1
    c) -1
    d) depends on compiler
    Ans: a

27. Correct declaration of array?
    a) int arr;
    b) int arr[10];
    c) arr int[10];
    d) int[10] arr;
    Ans: b

28. Array elements are stored in:
    a) Random order
    b) Sequential memory locations
    c) Linked list
    d) Binary tree
    Ans: b

29. Accessing out-of-bound index results in:
    a) Error
    b) Garbage value
    c) Crash
    d) Compiler warning
    Ans: b

30. A two-dimensional array is also called:
    a) Table
    b) Matrix
    c) Grid

d) All of the above
Ans: d

31. Size of int arr[10] on 32-bit compiler?
    a) 10 bytes
    b) 20 bytes
    c) 40 bytes
    d) 4 bytes
    Ans: c

32. sizeof(char[5]) = ?
    a) 1
    b) 5
    c) 10
    d) 0
    Ans: b

33. Which loop is best for array traversal?
    a) while
    b) for
    c) do-while
    d) switch
    Ans: b

34. Which statement is correct?
    a) Array size must be constant at compile time
    b) Array size can be dynamic
    c) Array stores values randomly
    d) Array elements cannot be accessed by index
    Ans: a

35. Which array declaration is valid?
    a) int arr[0];
    b) int arr[-5];
    c) int arr[5];
    d) int arr[]; without initializer
    Ans: c

36. Formula of address calculation in 2D array?
    a) Base + (i+j)*Size*
    *b) Base + (i*Col + j)*Size*
    *c) Base + (i*Row + j)*Size*
    *d) Base + (Row*Col)*Size
    Ans: b

37. For Base=1000, Col=3, Size=2 bytes, address of A[1][2]?
    a) 1004
    b) 1006
    c) 1010

d) 1012
Ans: c

38. What is the correct way to initialize an array?
    a) int arr[3]={1,2,3};
    b) int arr[]={1,2,3};
    c) int arr[3]; arr={1,2,3};
    d) Both a & b
    Ans: d

39. An array of arrays is called:
    a) Multi-dimensional array
    b) Pointer
    c) Function
    d) Linked list
    Ans: a

40. Which statement is wrong about arrays?
    a) Array size cannot be changed at runtime
    b) Array elements are of same type
    c) Array size can be decreased dynamically
    d) Array provides random access
    Ans: c

41. Output:

```
int arr[5]={10,20,30,40,50};

printf("%d", arr[2]);
```

a) 20
b) 30
c) 40
d) 10
Ans: b

42. If int arr[2][2]={{1,2},{3,4}};, what is arr[1][0]?
    a) 1
    b) 2
    c) 3
    d) 4
    Ans: c

43. Which array initialization is invalid?
    a) int arr[3]={1,2};
    b) int arr[]={1,2,3};
    c) int arr[2]={1,2,3};
    d) int arr[5]={0};
    Ans: c

44. Which function is used to calculate size of an array?
    a) length(arr)
    b) sizeof(arr)/sizeof(arr[0])
    c) size(arr)
    d) count(arr)
    Ans: b

45. Which of the following accesses element at 2nd row, 3rd column in 2D array?
    a) arr[1][2]
    b) arr[2][3]
    c) arr[3][2]
    d) arr[2][1]
    Ans: a

F. Mixed Questions

46. Which operator is used in if condition?
    a) Assignment
    b) Relational
    c) Bitwise
    d) Logical only
    Ans: b

47. Which storage class gives highest speed?
    a) auto
    b) static
    c) extern
    d) register
    Ans: d

48. Which loop executes zero or more times?
    a) while
    b) for
    c) do-while
    d) both a & b
    Ans: d

49. Can switch have duplicate case values?
    a) Yes
    b) No
    c) Compiler dependent
    d) Only in C++
    Ans: b

50. Default return type of main()?
    a) void
    b) int
    c) char

d) float
Ans: b

51. Which keyword is used to define global variable across multiple files?
    a) auto
    b) extern
    c) global
    d) static
    Ans: b

52. Which storage class is default for functions?
    a) static
    b) extern
    c) auto
    d) register
    Ans: b

53. Can arrays store different data types?
    a) Yes
    b) No
    c) Sometimes
    d) Only in structures
    Ans: b

54. Which loop is most commonly used in arrays?
    a) do-while
    b) while
    c) for
    d) none
    Ans: c

55. In 2D array declaration int arr[3][4]; how many elements?
    a) 7
    b) 10
    c) 12
    d) 16
    Ans: c

56. Which one is not a valid storage class in C?
    a) auto
    b) static
    c) volatile
    d) register
    Ans: c (volatile is a qualifier, not a storage class)

57. What is the default value of uninitialized local array?
    a) 0
    b) garbage
    c) null

d) none
Ans: b

58. Which C feature allows decision control?
    a) Data types
    b) Loops
    c) Conditional statements
    d) Storage classes
    Ans: c

59. In C, what does sizeof(int[5]) return in 32-bit system?
    a) 10
    b) 20
    c) 40
    d) 4
    Ans: c

60. The maximum index of an array declared as int arr[50]; is:
    a) 49
    b) 50
    c) 51
    d) 48
    Ans: a