Introduction to C and C++ (MCQs 1–10)

1. Who developed the C language?
A) James Gosling
B) Dennis Ritchie
C) Bjarne Stroustrup
D) Guido van Rossum
Answer: B) Dennis Ritchie

---

2. C was developed at which organization?
A) Microsoft
B) Apple
C) Bell Labs
D) IBM
Answer: C) Bell Labs

---

3. Who developed C++?
A) Dennis Ritchie
B) Bjarne Stroustrup
C) Ken Thompson
D) Brian Kernighan
Answer: B) Bjarne Stroustrup

---

4. C++ is an extension of which language?
A) Java
B) Pascal
C) C
D) Python
Answer: C) C

---

5. Procedural programming mainly focuses on:
A) Objects
B) Functions
C) Classes
D) Interfaces
Answer: B) Functions

---

6. Object-Oriented Programming mainly focuses on:
A) Functions
B) Procedures
C) Objects
D) Variables
Answer: C) Objects

---

7. Which function is mandatory in a C++ program?
A) start()
B) main()
C) init()
D) program()
Answer: B) main()

---

8. The execution of a C++ program starts from:
A) First line
B) Last line
C) main() function
D) Header file
Answer: C) main() function

---

9. Which command is used to compile a C++ program (GCC compiler)?
A) run program.cpp
B) compile program.cpp
C) g++ program.cpp
D) execute program.cpp
Answer: C) g++ program.cpp

---

10. The file extension of a C++ program is generally:
A) .c
B) .cpp
C) .java
D) .py
Answer: B) .cpp

✅ Data Types, Variables, Constants, Operators & I/O (MCQs 11–30)

11. Which of the following is a valid data type in C/C++?
A) integer
B) real
C) int
D) number
Answer: C) int

---

12. Which keyword is used to define a constant in C++?
A) const
B) constant
C) define
D) fixed
Answer: A) const

---

13. Which of the following is NOT a valid variable name?
A) total
B) _value
C) 2num
D) marks1
Answer: C) 2num

---

14. Scope of a local variable is limited to:
A) Entire program
B) Header file
C) Function in which it is declared
D) Main function only
Answer: C) Function in which it is declared

---

15. Which operator is used for modulus?
A) /
B) %
C) *

D) //
Answer: B) %

---

16. Which operator is logical AND?
A) &
B) &&
C) ||
D) !
Answer: B) &&

---

17. Bitwise AND operator is:
A) &&
B) &
C) ||
D) |
Answer: B) &

---

18. Size of int in most modern systems is:
A) 1 byte
B) 2 bytes
C) 4 bytes
D) 8 bytes
Answer: C) 4 bytes

---

19. Which function is used for formatted output in C?
A) scanf()
B) printf()
C) cin
D) cout
Answer: B) printf()

---

20. Which operator is used with cout?
A) >>
B) <<
C) ==

D) =
Answer: B) <<

---

21. Which operator is used with cin?
A) <<
B) >>
C) =
D) ==
Answer: B) >>

---

22. getchar() is used for:
A) Writing character
B) Reading character
C) Reading string
D) Writing string
Answer: B) Reading character

---

23. putchar() is used for:
A) Reading character
B) Writing character
C) Reading integer
D) Writing integer
Answer: B) Writing character

---

24. Header file for printf() and scanf() is:
A) iostream.h
B) stdio.h
C) conio.h
D) math.h
Answer: B) stdio.h

---

25. Header file for cin and cout is:
A) stdio.h
B) math.h
C) iostream

D) conio.h
Answer: C) iostream

---

26. Which is a unary operator?
A) +
B) -
C) ++
D) *
Answer: C) ++

---

27. Type casting means:
A) Changing variable name
B) Converting one data type into another
C) Defining variable
D) Deleting variable
Answer: B) Converting one data type into another

---

28. Single line comment in C++ starts with:
A) /*
B) //
C) #
D) --
Answer: B) //

---

29. Multi-line comment starts with:
A) //
B) ##
C) /*
D) <!--
Answer: C) /*

---

30. Which of the following is a keyword?
A) main
B) printf
C) int

D) sum
Answer: C) int

✅ Expressions, Conditional & Iterative Statements (MCQs 31–50)

31. Which operator has highest precedence?
A) +
B) *
C) =
D) ()
Answer: D) ()

---

32. If x = 5, what is value of x++?
A) 6
B) 5
C) 4
D) 10
Answer: B) 5

---

33. Pre-increment operator is:
A) x++
B) ++x
C) x--
D) --x
Answer: B) ++x

---

34. Which statement is used for decision making?
A) for
B) while
C) if
D) goto
Answer: C) if

---

35. switch statement is used for:
A) Looping
B) Multiple selection
C) Object creation
D) Function call
Answer: B) Multiple selection

---

36. Default case in switch is:
A) Optional
B) Mandatory
C) Illegal
D) None
Answer: A) Optional

---

37. Which loop checks condition first?
A) do-while
B) while
C) switch
D) if
Answer: B) while

---

38. Which loop executes at least once?
A) while
B) for
C) do-while
D) if
Answer: C) do-while

---

39. for loop is generally used when:
A) Iterations unknown
B) Iterations known
C) No iteration
D) Only condition
Answer: B) Iterations known

---

40. break statement is used to:
A) Continue loop
B) Exit loop
C) Restart loop
D) Skip iteration
Answer: B) Exit loop

---

41. continue statement is used to:
A) Exit loop
B) Skip current iteration
C) Stop program
D) Restart program
Answer: B) Skip current iteration

---

42. Nested loop means:
A) Loop inside loop
B) Loop outside function
C) Function inside loop
D) Loop in header
Answer: A) Loop inside loop

---

43. Relational operator for equality is:
A) =
B) ==
C) !=
D) >=
Answer: B) ==

---

44. Logical OR operator is:
A) |
B) ||
C) &
D) &&
Answer: B) ||

---

45. Expression 5 > 3 && 2 < 4 gives:
A) 0
B) 1
C) 2
D) Error
Answer: B) 1

---

46. Which is an infinite loop?
A) while(0)
B) while(1)
C) for(i=0;i<10;i++)
D) if(1)
Answer: B) while(1)

---

47. The ternary operator is:
A) ?:
B) ::
C) ??
D) %%
Answer: A) ?:

---

48. Which statement can replace multiple if-else?
A) while
B) for
C) switch
D) do
Answer: C) switch

---

49. A compound statement is enclosed within:
A) ()
B) []
C) {}
D) <>
Answer: C) {}

---

50. The condition in an if statement must return:
A) String
B) Float
C) Boolean/True-False value
D) Character
Answer: C) Boolean/True-False value

Functions (Q.51–70)

51. What is the main utility of functions in C/C++?
A) Increase program length
B) Code reusability and modularity
C) Reduce memory
D) Avoid loops
Answer: B) Code reusability and modularity

---

52. In Call by Value, changes made inside function:
A) Affect original variable
B) Do not affect original variable
C) Delete variable
D) Create error
Answer: B) Do not affect original variable

---

53. In Call by Reference, arguments are passed using:
A) Values
B) Constants
C) Addresses/References
D) Operators
Answer: C) Addresses/References

---

54. Which symbol is used for Call by Reference in C++?
A) *
B) &
C) %
D) #
Answer: B) &

---

55. A function that does not return any value is declared using:
A) int
B) float
C) void
D) return
Answer: C) void

---

56. A function returning integer must have return type:
A) void
B) int
C) char
D) bool
Answer: B) int

---

57. Inline functions are used to:
A) Increase execution time
B) Reduce function call overhead
C) Create arrays
D) Define structure
Answer: B) Reduce function call overhead

---

58. Keyword used for inline function is:
A) define
B) inline
C) static
D) const
Answer: B) inline

---

59. Which part of function specifies return type?
A) Function body
B) Function call
C) Function declaration
D) Header file
Answer: C) Function declaration

---

60. Function parameters are also called:
A) Local variables
B) Formal arguments
C) Constants
D) Operators
Answer: B) Formal arguments

---

61. Arguments passed during function call are called:
A) Formal parameters
B) Actual arguments
C) Static variables
D) Constants
Answer: B) Actual arguments

---

62. Function declaration is also known as:
A) Function body
B) Function prototype
C) Function loop
D) Function class
Answer: B) Function prototype

---

63. Function definition contains:
A) Only return type
B) Only parameters
C) Actual code/body of function
D) Only declaration
Answer: C) Actual code/body of function

---

64. Command line arguments are received by which function?
A) start()
B) main()
C) init()
D) display()
Answer: B) main()

---

65. Syntax for command line arguments is:
A) main()
B) main(void)
C) main(int argc, char *argv[])
D) void main()
Answer: C) main(int argc, char *argv[])

---

66. argc represents:
A) Argument count
B) Array count
C) Character count
D) Argument code
Answer: A) Argument count

---

67. argv is:
A) Integer variable
B) Array of strings
C) Float array
D) Structure
Answer: B) Array of strings

---

68. Function with variable number of arguments uses header file:
A) stdio.h
B) stdlib.h
C) stdarg.h
D) string.h
Answer: C) stdarg.h

---

69. Which macro is used to access variable arguments?
A) va_start
B) va_begin
C) va_init
D) va_endarg
Answer: A) va_start

---

70. Recursive function is a function that:
A) Has no return
B) Calls itself
C) Has no parameters
D) Has multiple loops
Answer: B) Calls itself

✅ One-Dimensional & Multi-Dimensional Arrays (Q.71–90)

71. An array stores:
A) Different data types
B) Same type of elements
C) Only integers
D) Only characters
Answer: B) Same type of elements

---

72. Correct declaration of integer array is:
A) int arr;
B) int arr[10];
C) array int arr;
D) int[10] arr;
Answer: B) int arr[10];

---

73. Array index in C/C++ starts from:
A) 1
B) 0
C) -1
D) 2
Answer: B) 0

---

74. Size of array int arr[5]; is:
A) 4 bytes
B) 5 bytes
C) 20 bytes (assuming 4 bytes int)

D) 10 bytes
Answer: C) 20 bytes

---

75. To access 3rd element of arr, index used is:
A) arr[3]
B) arr[2]
C) arr[1]
D) arr[0]
Answer: B) arr[2]

---

76. Array elements are stored in:
A) Random order
B) Contiguous memory
C) Stack only
D) File
Answer: B) Contiguous memory

---

77. Which loop is best for traversing array?
A) if
B) switch
C) for
D) goto
Answer: C) for

---

78. Character array is also known as:
A) Structure
B) String
C) Union
D) Integer array
Answer: B) String

---

79. String in C ends with:
A) \t
B) \0
C) \n

D) EOF
Answer: B) \0

---

80. Float array is declared as:
A) float arr[5];
B) int arr[5];
C) char arr[5];
D) double arr;
Answer: A) float arr[5];

---

81. Two-dimensional array is declared as:
A) int arr[][];
B) int arr[3][3];
C) int arr;
D) int[3,3] arr;
Answer: B) int arr[3][3];

---

82. In int arr[2][3], total elements are:
A) 5
B) 6
C) 3
D) 2
Answer: B) 6

---

83. arr[1][2] represents:
A) Row 1 Column 2
B) Row 2 Column 1
C) Row 0 Column 1
D) Error
Answer: A) Row 1 Column 2

---

84. Multi-dimensional array means:
A) One index
B) Two index only
C) More than one index

D) No index
Answer: C) More than one index

---

85. 3D array declaration example:
A) int arr[2][2][2];
B) int arr[2][2];
C) int arr[2];
D) int arr;
Answer: A) int arr[2][2][2];

---

86. Array name represents:
A) Last element
B) First element value
C) Base address
D) Size
Answer: C) Base address

---

87. Passing array to function actually passes:
A) Copy of array
B) Address of array
C) Single element
D) Nothing
Answer: B) Address of array

---

88. Which function is used to copy strings?
A) strcpy()
B) strlen()
C) strcat()
D) strcmp()
Answer: A) strcpy()

---

89. strlen() returns:
A) Size in bytes
B) Length of string
C) Index

D) Address
Answer: B) Length of string

---

90. Accessing array beyond its size causes:
A) Correct output
B) Compilation error
C) Runtime error/Undefined behavior
D) Warning only
Answer: C) Runtime error/Undefined behavior

✓ Structures and Unions (Q.91–100)

91. Structure is declared using keyword:
A) class
B) struct
C) union
D) define
Answer: B) struct

---

92. Structure can contain:
A) Same data types only
B) Different data types
C) Only integers
D) Only arrays
Answer: B) Different data types

---

93. Members of structure are accessed using:
A) :
B) ->
C) .
D) ::
Answer: C) .

---

94. Union is declared using keyword:
A) struct

B) union
C) class
D) typedef
Answer: B) union

---

95. In union, memory is shared by:
A) Each member separately
B) All members
C) Only first member
D) Only integer member
Answer: B) All members

---

96. Size of union is equal to:
A) Sum of members
B) Smallest member
C) Largest member
D) Average size
Answer: C) Largest member

---

97. Array of structures is declared as:
A) struct student s[10];
B) struct s student[10];
C) student struct[10];
D) array student struct;
Answer: A) struct student s[10];

---

98. Structure can be passed to function:
A) Yes
B) No
C) Only in C++
D) Only in C
Answer: A) Yes

---

99. Structure can be returned from function:
A) Yes
B) No

C) Only int
D) Only void
Answer: A) Yes

---

100. Structure containing union is:
A) Invalid
B) Allowed
C) Error
D) Deprecated
Answer: B) Allowed

Pointers and References in C++ (Q1–30)

1. A pointer variable stores:
A) Value
B) Address
C) Data type
D) Operator
Answer: B) Address

---

2. Which symbol is used to declare a pointer?
A) &
B) *
C) %
D) #
Answer: B) *

---

3. Dereferencing a pointer means:
A) Getting address
B) Accessing value stored at address
C) Deleting pointer
D) Assigning NULL
Answer: B) Accessing value stored at address

**4. If int x=10; int *p=&x; then p gives:*
A) Address of x
B) 10

C) Garbage
D) Error
Answer: B) 10

---

5. NULL pointer means:
A) Points to 0
B) Points to valid memory
C) Points to nothing
D) Points to itself
Answer: C) Points to nothing

---

6. Pointer to pointer is declared as:
A) int *p;
B) int *p;
C) int &p;
D) int p;
Answer: B) int **p;

---

7. If int **pp; then pp stores:
A) Value
B) Address of int
C) Address of pointer
D) Error
Answer: C) Address of pointer

---

8. Structure pointer member accessed using:
A) .
B) ->
C) ::
D) &
Answer: B) ->

---

9. Dangling pointer occurs when:
A) Pointer not initialized
B) Pointer points to freed memory
C) Pointer is NULL

D) Pointer is global
Answer: B) Pointer points to freed memory

---

10. Passing pointer to function allows:
A) Call by value
B) Modify original variable
C) No return
D) Error
Answer: B) Modify original variable

---

11. Returning pointer from function is valid if:
A) Returning local variable address
B) Returning static/dynamic memory address
C) Returning NULL only
D) Never allowed
Answer: B) Returning static/dynamic memory address

---

12. Array name represents:
A) Value
B) Address of first element
C) Size
D) Index
Answer: B) Address of first element

---

13. arr[i] is equivalent to:
A) *(arr+i)
B) arr+i
C) *arr+i
D) &arr
Answer: A) *(arr+i)

---

14. When passing array to function, actually passed:
A) Entire array
B) First element only
C) Base address

D) Copy
Answer: C) Base address

---

15. Reference variable is declared using:
A) *
B) &
C) %
D) @
Answer: B) &

---

16. Reference must be initialized at:
A) End
B) Declaration
C) Function call
D) Runtime
Answer: B) Declaration

---

17. Reference once initialized can:
A) Change reference
B) Refer to another variable
C) Cannot change binding
D) Be NULL
Answer: C) Cannot change binding

---

18. Reference is an alias for:
A) Pointer
B) Variable
C) Function
D) Class
Answer: B) Variable

---

19. Pointer arithmetic increments by:
A) 1 byte
B) Size of data type
C) 2 bytes

D) Random
Answer: B) Size of data type

---

20. Which is safer?
A) Pointer
B) Reference
C) Array
D) Macro
Answer: B) Reference

---

21. Pointer can be:
A) NULL
B) Reassigned
C) Both A and B
D) None
Answer: C) Both A and B

---

22. Reference cannot be:
A) Initialized
B) NULL
C) Used in function
D) Returned
Answer: B) NULL

---

23. sizeof(pointer) returns:
A) Size of value
B) Size of address
C) 0
D) Error
Answer: B) Size of address

---

24. Memory leak occurs when:
A) Not freeing allocated memory
B) Freeing twice
C) Using stack

D) Using static
Answer: A) Not freeing allocated memory

---

25. delete is used to:
A) Allocate memory
B) Free dynamic memory
C) Define pointer
D) Assign pointer
Answer: B) Free dynamic memory

---

26. new operator returns:
A) Value
B) Address
C) Reference
D) Error
Answer: B) Address

---

27. Pointer to structure declared as:
A) struct s *ptr;
B) struct *s ptr;
C) ptr struct s;
D) struct ptr s;
Answer: A) struct s *ptr;

---

28. Dereferencing NULL pointer causes:
A) Safe output
B) Compile error
C) Runtime error
D) Warning only
Answer: C) Runtime error

**29. p++ means:*
A) *(p++)
B) (*p)++
C) Error

D) None
Answer: A) *(p++)

---

30. Returning local variable address is:
A) Safe
B) Undefined behavior
C) Good practice
D) Required
Answer: B) Undefined behavior

✓ Memory Allocation in C++ (Q31–45)

31. Static memory allocated at:
A) Runtime
B) Compile time
C) Execution end
D) Function call
Answer: B) Compile time

---

32. Dynamic memory allocated at:
A) Compile time
B) Runtime
C) Preprocessing
D) Linking
Answer: B) Runtime

---

33. malloc() returns:
A) int
B) void*
C) char
D) bool
Answer: B) void*

---

34. calloc() initializes memory to:
A) 1

B) Garbage
C) 0
D) NULL
Answer: C) 0

---

35. free() is used with:
A) new
B) malloc/calloc
C) static
D) reference
Answer: B) malloc/calloc

---

36. delete is paired with:
A) malloc
B) calloc
C) new
D) free
Answer: C) new

---

37. new[] is used for:
A) Single variable
B) Array allocation
C) Free memory
D) Pointer arithmetic
Answer: B) Array allocation

---

38. delete[] is used for:
A) Single delete
B) Array delete
C) Reference
D) Stack
Answer: B) Array delete

---

39. Stack memory is:
A) Dynamic
B) Static

C) Automatic
D) Manual
Answer: C) Automatic

---

40. Heap memory is:
A) Stack
B) Static
C) Dynamic
D) Register
Answer: C) Dynamic

---

41. Memory allocated by new must be released by:
A) free
B) delete
C) malloc
D) return
Answer: B) delete

---

42. Static variables stored in:
A) Heap
B) Stack
C) Data segment
D) Cache
Answer: C) Data segment

---

43. Global variables stored in:
A) Heap
B) Stack
C) Data segment
D) Register
Answer: C) Data segment

---

44. malloc() requires header:
A) iostream
B) stdlib.h
C) fstream

D) string
Answer: B) stdlib.h

---

45. new operator in C++ is:
A) Function
B) Keyword
C) Macro
D) Class
Answer: B) Keyword

✓ File I/O & Preprocessor (Q46–65)

46. Header for file handling in C++:
A) stdio.h
B) fstream
C) iostream
D) string
Answer: B) fstream

---

47. ifstream is used for:
A) Writing file
B) Reading file
C) Deleting file
D) Renaming file
Answer: B) Reading file

---

48. ofstream is used for:
A) Reading
B) Writing
C) Both
D) None
Answer: B) Writing

---

49. fstream is used for:
A) Only read

B) Only write
C) Both read and write
D) None
Answer: C) Both read and write

---

50. get() reads:
A) String
B) Character
C) Integer
D) File size
Answer: B) Character

---

51. put() writes:
A) String
B) Character
C) Float
D) Line
Answer: B) Character

---

52. read() used for:
A) Text only
B) Binary data
C) Console
D) Macro
Answer: B) Binary data

---

53. write() used for:
A) Text
B) Binary
C) Console
D) Macro
Answer: B) Binary

---

54. Random file access uses:
A) seekg()
B) close()

C) open()
D) get()
Answer: A) seekg()

---

55. #include is used for:
A) Define macro
B) Include header
C) Error
D) Loop
Answer: B) Include header

---

56. #define is used for:
A) Include file
B) Define macro
C) Error
D) Loop
Answer: B) Define macro

---

57. #ifdef checks:
A) If variable defined
B) If macro defined
C) If file open
D) If class
Answer: B) If macro defined

---

58. #ifndef means:
A) If defined
B) If not defined
C) If file open
D) If error
Answer: B) If not defined

---

59. #undef is used to:
A) Define macro
B) Remove macro
C) Include file

D) Throw error
Answer: B) Remove macro

---

60. #error generates:
A) Runtime error
B) Compile-time error
C) Logical error
D) Warning
Answer: B) Compile-time error

---

61. #if used for:
A) Loop
B) Conditional compilation
C) Function
D) Class
Answer: B) Conditional compilation

---

62. #elif stands for:
A) Else if
B) End if
C) Error
D) Define
Answer: A) Else if

---

63. #endif closes:
A) Loop
B) If block
C) Conditional directive
D) Class
Answer: C) Conditional directive

---

64. Macros are processed by:
A) Compiler
B) Linker
C) Preprocessor

D) Loader
Answer: C) Preprocessor

---

65. File close method is:
A) end()
B) close()
C) stop()
D) exit()
Answer: B) close()

✅ Classes, Overloading, Inheritance, Polymorphism, Exception (Q66–100)

66. OOP stands for:
A) Object Oriented Programming
B) Operator Oriented Programming
C) Only Object Program
D) None
Answer: A

---

67. Class keyword is:
A) object
B) class
C) struct
D) define
Answer: B

---

68. Constructor has same name as:
A) Function
B) Variable
C) Class
D) Object
Answer: C

---

69. Constructor has:
A) Return type
B) No return type
C) int return
D) void return
Answer: B

---

70. Copy constructor takes:
A) int
B) Same class object reference
C) float
D) pointer
Answer: B

---

71. Function overloading means:
A) Same function name different parameters
B) Different name same parameters
C) Same class
D) Same return only
Answer: A

---

72. Operator overloading allows:
A) Change operator meaning
B) Delete operator
C) Replace compiler
D) Stop program
Answer: A

---

73. Overloaded assignment operator is:
A) +
B) -
C) =
D) *
Answer: C

---

74. Inheritance allows:
A) Code reuse
B) Memory leak
C) Error
D) None
Answer: A

---

75. Multiple inheritance means:
A) One parent
B) Two or more parents
C) No parent
D) No child
Answer: B

---

76. Multilevel inheritance means:
A) A→B→C
B) A,B→C
C) Only A
D) None
Answer: A

---

77. Virtual function supports:
A) Compile-time binding
B) Runtime polymorphism
C) Static binding
D) Overloading
Answer: B

---

78. Pure virtual function declared using:
A) =0
B) =1
C) void
D) NULL
Answer: A

---

79. Class with pure virtual function is:
A) Normal class
B) Abstract class
C) Static class
D) Final class
Answer: B

---

80. throw is used to:
A) Catch exception
B) Generate exception
C) Ignore exception
D) End program
Answer: B

---

81. catch is used to:
A) Throw error
B) Handle exception
C) Define class
D) Loop
Answer: B

---

82. Multiple catch blocks handle:
A) One exception
B) Different exceptions
C) No exception
D) Only int
Answer: B

---

83. catch(...) catches:
A) Specific
B) All exceptions
C) No exception
D) Only float
Answer: B

---

84. Rethrowing exception uses:
A) throw;
B) catch;
C) error;
D) retry;
Answer: A

---

85. Protected members accessible in:
A) Same class
B) Derived class
C) Outside
D) None
Answer: B

---

86. Private members accessible in:
A) Same class only
B) Derived class
C) Outside
D) All
Answer: A

---

87. Template class defined using:
A) template<class T>
B) class<T>
C) define T
D) typedef T
Answer: A

---

88. Templates support:
A) Polymorphism
B) Generic programming
C) Inheritance
D) Exception
Answer: B

---

89. Operator+ overloaded as:
A) function
B) class
C) macro
D) variable
Answer: A

---

90. Object passed to function as:
A) Parameter
B) Operator
C) Macro
D) File
Answer: A

---

91. Destructor called when:
A) Object created
B) Object destroyed
C) Program start
D) Function call
Answer: B

---

92. Destructor name is:
A) ~ClassName
B) !ClassName
C) -ClassName
D) deleteClass
Answer: A

---

93. Constructor overloading means:
A) Multiple constructors
B) One constructor
C) No constructor
D) Static constructor
Answer: A

---

94. Static class variable shared by:
A) One object
B) All objects
C) None
D) Derived only
Answer: B

---

95. Polymorphism means:
A) Many forms
B) One form
C) No form
D) Structure
Answer: A

---

96. Base class accessed using:
A) : public Base
B) public:Base
C) class Base
D) Base()
Answer: A

---

97. Exception handling improves:
A) Security
B) Robustness
C) Compilation
D) Linking
Answer: B

---

98. Dynamic binding achieved by:
A) Static function
B) Virtual function
C) Macro
D) Template
Answer: B

---

99. Diamond problem occurs in:
A) Single inheritance
B) Multiple inheritance
C) Multilevel
D) None
Answer: B

---

100. Virtual base class solves:
A) Memory leak
B) Diamond problem
C) Exception
D) Overloading
Answer: B

Unit 1: Introduction to C and C++ (Q1–15)

1. Who developed C language?
Dennis Ritchie.

2. Where was C developed?
At Bell Laboratories.

3. Who developed C++?
Bjarne Stroustrup.

4. C++ is an extension of which language?
C language.

5. What is procedural programming?
Programming based on functions and procedures.

6. What is Object-Oriented Programming (OOP)?
Programming based on objects and classes.

7. What is the entry point of a C++ program?
main() function.

8. What is the general syntax of main() in C++?
int main() { }

9. What does a compiler do?
Converts source code into machine code.

10. What is the extension of C++ file?
.cpp

11. What is linking?
Process of combining object files into executable file.

12. What is source code?
Program written in high-level language.

13. What is executable file?
Machine code file that can be run.

14. What is compilation error?
Error detected during compilation.

15. What is syntax error?
Error due to incorrect grammar of language.

✅ Unit 2: Data Types, Variables, Constants, Operators & Basic I/O (Q16–40)

16. What is a variable?
A named memory location.

17. What is a constant?
A value that does not change during execution.

18. Which keyword is used to declare constant in C++?
const

19. What is data type?
Specifies type of data stored in variable.

20. Name two basic data types.
int, float.

21. What is size of int (generally)?
4 bytes.

22. What is scope of variable?
Region where variable is accessible.

23. What is local variable?
Declared inside function.

24. What is global variable?
Declared outside all functions.

25. What is type casting?
Converting one data type to another.

26. What are arithmetic operators?
+, -, *, /, %.

27. What are logical operators?
&&, ||, !

28. What are bitwise operators?
&, |, ^, ~, <<, >>

29. What is increment operator?
++

30. What is decrement operator?

31. What is single-line comment in C++?
// comment

32. What is multi-line comment?
/* comment */

33. Which function prints output in C?
printf()

34. Which function takes input in C?
scanf()

35. Which object is used for output in C++?
cout

36. Which object is used for input in C++?
cin

37. Which operator is used with cout?
<<

38. Which operator is used with cin?

39. Header file for printf()?
stdio.h

40. Header file for cin and cout?
iostream

✅ Unit 3: Expressions, Conditional & Iterative Statements (Q41–60)

41. What is an expression?
Combination of variables, operators and values.

42. What is unary operator?
Operator working on one operand.

43. What is binary operator?
Operator working on two operands.

44. What determines order of evaluation in expressions?
Operator precedence.

45. Which operator has higher precedence: * or + ?
*

46. What is if statement used for?
Decision making.

47. Syntax of if statement?
if(condition) { }

48. What is switch statement?
Multi-way selection statement.

49. What is default in switch?
Optional block executed if no case matches.

50. What is loop?
Statement that repeats execution.

51. Name three loops in C++.
for, while, do-while.

52. Which loop executes at least once?
do-while loop.

53. What does break do?
Terminates loop or switch.

54. What does continue do?
Skips current iteration.

55. What is nested loop?
Loop inside another loop.

56. What is relational operator?
Operator comparing two values.

57. Example of relational operator?

58. What does && mean?
Logical AND.

59. What does || mean?
Logical OR.

60. What is infinite loop?
Loop that never ends.

✓ Unit 4: Functions (Q61–75)

61. What is function?
Block of code performing specific task.

62. Why functions are used?
Code reusability and modularity.

63. What is function prototype?
Function declaration.

64. What is function definition?
Actual implementation of function.

65. What is call by value?
Passing copy of variable.

66. What is call by reference?
Passing address/reference of variable.

67. What is return type?
Type of value returned by function.

68. What is void function?
Function that returns nothing.

69. What is inline function?
Function expanded at compile time.

70. Keyword for inline function?
inline

71. What are formal parameters?
Parameters in function definition.

72. What are actual parameters?
Arguments in function call.

73. What are command line arguments?
Arguments passed to main() during execution.

74. Syntax for command line arguments?
int main(int argc, char *argv[])

75. Which header is used for variable arguments?
stdarg.h

✅ Unit 4: Arrays (Q76–100)

76. What is array?
Collection of same data type elements.

77. How to declare integer array?
int arr[10];

78. Array index starts from?
0

79. How to access array element?
arr[index]

80. How many elements in int arr[5]?
5

81. What is array initialization?
Assigning values at declaration.

82. Example of array initialization?
int arr[3]={1,2,3};

83. How to traverse array?
Using loop.

84. What is character array?
Array of char elements.

85. What is string in C?
Character array ending with '\0'.

86. What is two-dimensional array?
Array with rows and columns.

87. Syntax of 2D array declaration?
int arr[3][3];

88. Total elements in arr[2][4]?
8

89. How to access 2D element?
arr[i][j]

90. What is multi-dimensional array?
Array with more than two dimensions.

91. Example of 3D array declaration?
int arr[2][2][2];

92. What is base address of array?
Address of first element.

93. Are array elements stored contiguously?
Yes.

94. Can arrays be passed to functions?
Yes.

95. What is size of int arr[4] (4 bytes int)?
16 bytes.

96. What happens if array index exceeds size?
Undefined behavior.

97. Which function finds string length?
strlen()

98. Which function copies string?
strcpy()

99. What is array of floats declared as?
float arr[5];

100. What is main advantage of array?
Stores multiple values under single name.

Derived Data Types – Structures and Unions (Q101–120)

101. What is a structure?
A user-defined data type that groups different data types.

102. Which keyword is used to declare a structure?
struct

103. What is the main utility of a structure?
To store related data of different types.

104. How are structure members accessed?
Using dot (.) operator.

105. How to declare a structure variable?
struct student s;

106. What is structure initialization?
Assigning values at declaration.

107. Can structures contain different data types?
Yes.

108. What is a union?
User-defined type where all members share same memory.

109. Which keyword is used to declare union?
union

110. What is size of union?
Size of largest member.

111. Can structure contain array?
Yes.

112. What is array of structures?
Collection of structure variables in array form.

113. How to declare array of structures?
struct student s[10];

114. Can structure be passed to function?
Yes.

115. Can structure be returned from function?
Yes.

116. What is nested structure?
Structure inside another structure.

117. Can union be member of structure?
Yes.

118. Can structure be member of union?
Yes.

119. How to access structure pointer member?
Using -> operator.

120. What is main difference between structure and union?
Structure allocates separate memory; union shares memory.

✅ Pointers and References in C++ (Q121–150)

121. What is pointer?
Variable that stores address of another variable.

122. How to declare pointer to int?
int *p;

123. What is dereferencing?
Accessing value using * operator.

124. What does & operator do?
Gives address of variable.

125. What is NULL pointer?
Pointer that points to nothing.

126. What is pointer to pointer?
Pointer storing address of another pointer.

127. How to declare pointer to pointer?
int **pp;

128. What is dangling pointer?
Pointer pointing to freed memory.

129. What is wild pointer?
Uninitialized pointer.

130. What is pointer arithmetic?
Operations on pointer addresses.

131. What does p++ do?
Moves pointer to next memory location.

132. What is array name equivalent to?
Base address of array.

133. Can arrays be passed to function?
Yes.

134. What is reference in C++?
Alias for a variable.

135. How to declare reference?
int &r = x;

136. Can reference be NULL?
No.

137. Can reference be reassigned?
No.

138. What is call by reference?
Passing reference to function.

139. Advantage of references over pointers?
Safer and easier to use.

140. What happens when dereferencing NULL?
Runtime error.

141. What is sizeof(pointer)?
Size of address.

142. What is pointer to structure?
Pointer storing address of structure variable.

143. How to access pointer to structure member?
Using ->

144. Can function return pointer?
Yes.

145. Should function return address of local variable?
No.

146. What is memory leak?
Unreleased allocated memory.

147. Which operator allocates memory in C++?
new

148. Which operator frees memory in C++?
delete

149. What is difference between pointer and reference?
Pointer stores address; reference is alias.

150. Can pointer change its address?
Yes.

✅ Memory Allocation in C++ (Q151–165)

151. What is static memory allocation?
Memory allocated at compile time.

152. What is dynamic memory allocation?
Memory allocated at runtime.

153. Where is dynamic memory allocated?
Heap.

154. Where are local variables stored?
Stack.

155. Which function allocates memory in C?
malloc()

156. Which function initializes memory to zero?
calloc()

157. Which function frees memory in C?
free()

158. What does new return?
Address of allocated memory.

159. What is delete[] used for?
Freeing array allocated with new[].

160. What is heap?
Memory used for dynamic allocation.

161. What is stack?
Memory used for local variables.

162. What happens if memory not freed?
Memory leak.

163. Which header file for malloc()?
stdlib.h

164. Is new safer than malloc()?
Yes.

165. What is difference between malloc and new?
new initializes and is C++ operator; malloc does not.

✅ File I/O & Preprocessor Directives (Q166–185)

166. Which header is used for file handling in C++?
fstream

167. Which class reads file?
ifstream

168. Which class writes file?
ofstream

169. Which class reads and writes file?
fstream

170. How to open file?
open()

171. How to close file?
close()

172. What does get() do?
Reads character.

173. What does put() do?
Writes character.

174. What is read()?
Reads binary data.

175. What is write()?
Writes binary data.

176. What is random access in file?
Accessing any position directly.

177. Which function used for random access?
seekg()

178. What is #include?
Preprocessor directive to include header.

179. What is #define?
Defines macro.

180. What is macro?
Symbolic constant.

181. What is #ifdef?
Checks if macro defined.

182. What is #ifndef?
Checks if macro not defined.

183. What is #undef?
Removes macro definition.

184. What is #error?
Generates compile-time error.

185. What is #endif?
Ends conditional directive.

✅ Classes, Overloading, Inheritance, Polymorphism & Exception Handling (Q186–200)

186. What is class?
Blueprint for objects.

187. What is object?
Instance of class.

188. What is constructor?
Special function to initialize object.

189. Does constructor have return type?
No.

190. What is constructor overloading?
Multiple constructors with different parameters.

191. What is copy constructor?
Constructor that copies object.

192. What is function overloading?
Same function name with different parameters.

193. What is operator overloading?
Giving new meaning to operators.

194. What is inheritance?
Deriving new class from existing class.

195. What is multiple inheritance?
Class derived from more than one base class.

196. What is multilevel inheritance?
Inheritance chain A→B→C.

197. What is polymorphism?
Ability to take many forms.

198. What is virtual function?
Function supporting runtime polymorphism.

199. What is exception handling?
Handling runtime errors using try-catch.

200. What is throw keyword?
Used to generate exception.

Unit 1: Introduction to C and C++

---

1. Explain the history of C and C++.

Answer:
C was developed by Dennis Ritchie in 1972 at Bell Labs. It was designed for system programming. C++ was developed by Bjarne Stroustrup in 1985 as an extension of C to support Object-Oriented Programming concepts like classes and objects.

---

2. Differentiate between Procedural and Object-Oriented Programming.

Answer:
Procedural programming focuses on functions and step-by-step instructions (e.g., C).
Object-Oriented Programming focuses on objects, classes, inheritance, and encapsulation (e.g., C++).

---

3. Write a simple C++ program to print "Hello World".

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World";
    return 0;
}
```

---

4. Explain the structure of a C++ program.

Answer:
A C++ program generally contains:

- Header files
- main() function
- Variable declarations
- Statements
- Return statement

---

5. Explain compilation and execution steps in C++.

Answer:

1. Write source code (.cpp file)
2. Compile using compiler (e.g., g++)
3. Generate executable file

4. Run executable

✓ Unit 2: Data Types, Variables & I/O

---

6. Define variable and give example.

Answer:
A variable is a named memory location.
Example:

```
int age = 20;
```

---

7. Explain data types in C++.

Answer:
Data types define type of data stored in variable such as int, float, char, double.

---

8. Write a program to add two numbers using cin and cout.

```
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << "Sum = " << a + b;
    return 0;
}
```

---

9. Explain scope of variables.

Answer:
Scope defines where variable can be accessed.
Types: Local scope and Global scope.

---

10. What are constants? Give example.

```
const float PI = 3.14;
```

---

11. Explain type casting with example.

```
int a = 5;
float b = (float)a;
```

---

12. Write a program using printf and scanf in C.

```
#include <stdio.h>
int main() {
    int num;
    scanf("%d", &num);
    printf("Number = %d", num);
    return 0;
}
```

---

13. Explain arithmetic operators with example.

Answer:
Operators: +, -, *, /, %
Example:

```
int x = 10 % 3;  // Output 1
```

---

14. Explain logical operators.

Answer:
&& (AND), || (OR), ! (NOT)

---

15. Explain bitwise operators.

Answer:
&, |, ^, ~, <<, >> operate at bit level.

---

16. Write a program to swap two numbers without third variable.

```
int a = 5, b = 3;
a = a + b;
b = a - b;
a = a - b;
```

---

17. Explain comments in C++.

Answer:
Single-line: //
Multi-line: /* */

18. Write a program to read and print a character using getchar().

```
#include <stdio.h>
int main() {
    char ch;
    ch = getchar();
    putchar(ch);
    return 0;
}
```

---

19. What is keyword? Give examples.

Answer:
Reserved words like int, float, if, while.

---

20. Explain increment operators.

Answer:
Pre-increment (++x)
Post-increment (x++)

✅ Unit 3: Expressions & Control Statements

---

21. Explain operator precedence.

Answer:
Operators are evaluated based on priority (* before +).

---

22. Write program to check even or odd.

```
int n;
cin >> n;
if(n % 2 == 0)
    cout << "Even";
else
    cout << "Odd";
```

---

23. Write a program using switch to print day name.

```
int d;
cin >> d;
switch(d) {
    case 1: cout<<"Monday"; break;
    case 2: cout<<"Tuesday"; break;
}
```

---

24. Explain while loop with example.

```
int i=1;
while(i<=5) {
    cout<<i;
    i++;
}
```

---

25. Explain do-while loop.

Executes at least once.

---

26. Write a program to print 1 to 10 using for loop.

```
for(int i=1;i<=10;i++)
    cout<<i;
```

---

27. What is break statement?

Terminates loop.

---

28. What is continue statement?

Skips current iteration.

---

29. Write nested loop example.

```
for(int i=1;i<=3;i++){
    for(int j=1;j<=3;j++)
        cout<<"*";
}
```

---

30. Write program to find factorial using loop.

```
int n,f=1;
cin>>n;
for(int i=1;i<=n;i++)
```

```
    f*=i;
cout<<f;
```

✓ Unit 4: Functions

---

31. Define function with example.

```
int add(int a,int b){
    return a+b;
}
```

---

32. Explain call by value.

Original variable not modified.

---

33. Explain call by reference with example.

```
void swap(int &a,int &b){
    int t=a;
    a=b;
    b=t;
}
```

---

34. What is void function?

Function that returns nothing.

---

35. Explain inline function.

```
inline int square(int x){
    return x*x;
}
```

---

36. Difference between declaration and definition.

Declaration: prototype
Definition: actual body.

---

37. Explain command line arguments.

```
int main(int argc,char *argv[])
```

38. Write recursive function to find factorial.

```
int fact(int n){
    if(n==0) return 1;
    return n*fact(n-1);
}
```

✓ Arrays

39. Define array with example.

```
int arr[5];
```

40. Write program to find sum of array elements.

```
int arr[5], sum=0;
for(int i=0;i<5;i++){
    cin>>arr[i];
    sum+=arr[i];
}
cout<<sum;
```

41. Explain 2D array declaration.

```
int arr[3][3];
```

42. Write program to print matrix.

```
int arr[2][2]={{1,2},{3,4}};
for(int i=0;i<2;i++){
    for(int j=0;j<2;j++)
        cout<<arr[i][j]<<" ";
}
```

43. What is multi-dimensional array?

Array with more than two dimensions.

44. Write program to search element in array.

```
int key,flag=0;
for(int i=0;i<5;i++){
    if(arr[i]==key)
        flag=1;
}
```

45. Write program to reverse array.

```
for(int i=4;i>=0;i--)
    cout<<arr[i];
```

---

46. What is string in C?

Character array ending with '\0'.

---

47. Write program to find length of string.

```
#include <cstring>
cout<<strlen("Hello");
```

---

48. What is base address of array?

Address of first element.

---

49. How are arrays passed to functions?

By passing base address.

---

50. Write function to find maximum element in array.

```
int max(int arr[],int n){
    int m=arr[0];
    for(int i=1;i<n;i++)
        if(arr[i]>m)
            m=arr[i];
    return m;
}
```

Unit 5: Derived Data Types – Structures & Unions

---

51. What is a structure? Explain with example.

Answer:
A structure is a user-defined data type that groups different data types under one name.

```
#include <iostream>
using namespace std;

struct Student {
    int roll;
    string name;
};

int main() {
    Student s1 = {1, "Rahul"};
    cout << s1.roll << " " << s1.name;
    return 0;
}
```

---

52. What is a union? How is it different from structure?

Answer:
Union allows different members to share the same memory location.

```
union Data {
    int i;
    float f;
};
```

Difference: Structure allocates separate memory; union shares memory.

---

53. Write a program to input and display structure data.

```
#include <iostream>
using namespace std;

struct Employee {
    int id;
    float salary;
};

int main() {
    Employee e;
    cin >> e.id >> e.salary;
    cout << e.id << " " << e.salary;
    return 0;
}
```

---

54. What is array of structures? Write example.

```
struct Student {
    int roll;
};

int main() {
    Student s[3];
}
```

---

55. Explain nested structure with example.

```
struct Address {
    string city;
};

struct Person {
    string name;
    Address addr;
};
```

---

56. How to pass structure to function?

```
void display(Student s){
    cout<<s.roll;
}
```

---

57. Write function returning structure.

```
Student create(){
    Student s={1,"Aman"};
    return s;
}
```

---

58. Explain structure pointer with example.

```
Student s1;
Student *ptr=&s1;
ptr->roll=10;
```

---

59. Can union be member of structure? Explain.

Yes, union can be inside structure to save memory.

---

60. Write example of structure containing union.

```
struct Info {
    int type;
    union {
        int i;
        float f;
```

```
    } data;
};
```

✅ Unit 6: Pointers and References

---

61. What is pointer? Give example.

```
int x=10;
int *p=&x;
```

---

62. Explain dereferencing.

```
cout<<*p;  // prints 10
```

---

63. What is pointer to pointer?

```
int **pp=&p;
```

---

64. Write program swapping using pointers.

```
void swap(int *a,int *b){
    int t=*a;
    *a=*b;
    *b=t;
}
```

---

65. What is dangling pointer?

Pointer pointing to freed memory.

---

66. Write program demonstrating array as pointer.

```
int arr[3]={1,2,3};
int *p=arr;
cout<<*(p+1);
```

---

67. Explain passing array to function.

```
void print(int arr[],int n){
    for(int i=0;i<n;i++)
        cout<<arr[i];
}
```

---

68. What is reference?

```
int x=5;
int &r=x;
```

---

69. Write program using call by reference.

```
void increment(int &a){
    a++;
}
```

---

70. Difference between pointer and reference?

Pointer can change address; reference cannot.

---

71. What happens if NULL pointer dereferenced?

Runtime error.

---

72. Write program returning pointer safely.

```
int* getStatic(){
    static int x=10;
    return &x;
}
```

---

73. What is pointer arithmetic?

```
int arr[3]={1,2,3};
int *p=arr;
p++;
```

---

74. Write pointer to structure example.

```
struct A{int x;};
A obj;
A *ptr=&obj;
ptr->x=5;
```

---

75. What is memory leak?

Not freeing allocated memory.

✅ Unit 7: Memory Allocation

---

76. What is dynamic memory allocation?

Memory allocated at runtime using new or malloc.

---

77. Write example using new and delete.

```
int *p=new int;
*p=20;
delete p;
```

---

78. Write example of dynamic array.

```
int *arr=new int[5];
delete[] arr;
```

---

79. Write C program using malloc and free.

```
#include <stdlib.h>
int *p=(int*)malloc(sizeof(int));
free(p);
```

---

80. Difference between static and dynamic memory?

Static allocated at compile time; dynamic at runtime.

✅ Unit 8: File I/O & Preprocessor

---

81. Write program to write file using ofstream.

```
#include <fstream>
ofstream file("data.txt");
file<<"Hello";
file.close();
```

---

82. Write program to read file using ifstream.

```
ifstream file("data.txt");
string s;
```

```
file>>s;
cout<<s;
```

---

83. What is fstream?

Class used for reading and writing files.

---

84. Explain random file access.

Using seekg() and seekp() to move file pointer.

---

85. Write example of seekg().

```
file.seekg(0, ios::beg);
```

---

86. What is #define? Give example.

```
#define PI 3.14
```

---

87. What is conditional compilation?

Using #if, #ifdef, etc.

---

88. Write example using #ifdef.

```
#define TEST
#ifdef TEST
cout<<"Defined";
#endif
```

---

89. What is macro?

Symbolic name representing value or code.

---

90. What is #undef?

Removes macro definition.

✅ Unit 9: Classes in C++

---

91. Define class with example.

```
class Student{
public:
    int roll;
};
```

---

92. What is constructor?

Special function to initialize object.

---

93. Write example of constructor.

```
class A{
public:
    A(){ cout<<"Constructor"; }
};
```

---

94. What is constructor overloading?

Multiple constructors with different parameters.

---

95. Write function overloading example.

```
int add(int a,int b){ return a+b; }
float add(float a,float b){ return a+b; }
```

---

96. What is copy constructor?

```
A(A &obj){}
```

---

97. Explain access specifiers.

public, private, protected.

---

98. What is static class variable?

Shared among all objects.

99. Write template class example.

```
template <class T>
class Test{
    T x;
};
```

---

100. Explain OOP principles.

Encapsulation, Abstraction, Inheritance, Polymorphism.

UNIT 1: Introduction to C and C++

---

1. Explain the history and evolution of C and C++. Discuss their features.

Answer:
C language was developed by Dennis Ritchie in 1972 at Bell Laboratories. It was designed for system programming and operating system development. C is structured and procedural in nature.

C++ was developed by Bjarne Stroustrup in 1985 as an extension of C. It added Object-Oriented Programming (OOP) features like classes, objects, inheritance, polymorphism, and encapsulation.

Features of C:

- Procedural programming
- Fast execution
- Low-level memory access
- Portable

Features of C++:

- Supports both procedural and OOP
- Encapsulation
- Inheritance
- Polymorphism
- Data abstraction

---

2. Explain the structure of a C++ program and write a simple example.

Answer:
A C++ program consists of:

1. Header files
2. Namespace declaration
3. main() function
4. Variable declarations
5. Statements
6. Return statement

```
#include <iostream>
using namespace std;

int main() {
    cout << "Welcome to C++";
    return 0;
}
```

---

3. Differentiate between Procedural Programming and Object-Oriented Programming.

Answer:

Procedural Programming	Object-Oriented Programming
Based on functions	Based on objects
Top-down approach	Bottom-up approach
Data is global	Data is hidden
Example: C	Example: C++

✅ UNIT 2: Data Types, Variables & I/O

---

4. Explain data types in C++ with suitable examples.

Answer:
Data types define the type of data a variable can hold.

- int → Integer values

- float → Decimal values
- char → Characters
- double → Large decimal values

```
int age = 20;
float marks = 89.5;
char grade = 'A';
```

---

5. Explain scope of variables with example.

Answer:
Scope defines accessibility of variable.

- Local variable → Inside function
- Global variable → Outside function

```
int x = 10; // Global

int main() {
    int y = 5; // Local
}
```

---

6. Explain operators in C++ with examples.

Answer:

Arithmetic: + - * / %
Logical: && || !
Relational: > < == !=

```
int a = 10, b = 5;
cout << a + b;
```

---

7. Write a program to perform arithmetic operations using switch statement.

```
#include <iostream>
using namespace std;

int main() {
    int a,b;
    char op;
    cin >> a >> b >> op;

    switch(op) {
        case '+': cout << a+b; break;
        case '-': cout << a-b; break;
        case '*': cout << a*b; break;
        case '/': cout << a/b; break;
    }
```

```
    return 0;
}
```

✓ UNIT 3: Expressions & Control Statements

---

8. Explain operator precedence and associativity.

Answer:
Operator precedence determines order of evaluation.
Example:
Multiplication has higher precedence than addition.

Associativity determines direction (left-to-right or right-to-left).

---

9. Write a program to check whether a number is prime.

```
#include <iostream>
using namespace std;

int main() {
    int n, count=0;
    cin >> n;
    for(int i=1;i<=n;i++){
        if(n%i==0)
            count++;
    }
    if(count==2)
        cout<<"Prime";
    else
        cout<<"Not Prime";
}
```

---

10. Explain different looping statements with examples.

Answer:
for → Fixed iterations
while → Entry-controlled
do-while → Exit-controlled

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

✓ UNIT 4: Functions & Arrays

---

11. Explain call by value and call by reference with program.

```
void swap(int &a,int &b){
    int t=a;
    a=b;
    b=t;
}
```

Call by reference modifies original values.

---

12. Explain recursive functions with example (Factorial).

```
int fact(int n){
    if(n==0)
        return 1;
    return n*fact(n-1);
}
```

---

13. Explain one-dimensional array with program to find sum.

```
int arr[5]={1,2,3,4,5};
int sum=0;
for(int i=0;i<5;i++)
    sum+=arr[i];
```

---

14. Explain two-dimensional array with matrix addition program.

```
int a[2][2]={{1,2},{3,4}};
int b[2][2]={{5,6},{7,8}};
int c[2][2];

for(int i=0;i<2;i++)
    for(int j=0;j<2;j++)
        c[i][j]=a[i][j]+b[i][j];
```

✓ UNIT 5: Structures & Unions

---

15. Explain structure with example.

```
struct Student{
    int roll;
    float marks;
};
```

16. Explain union and memory sharing concept.

Union shares same memory among members. Only one member holds value at a time.

✓ UNIT 6: Pointers & References

---

17. Explain pointer with program example.

```
int x=10;
int *p=&x;
cout<<*p;
```

---

18. Explain pointer arithmetic.

```
int arr[3]={1,2,3};
int *p=arr;
p++;
```

---

19. Differentiate between pointer and reference.

Pointer → stores address
Reference → alias of variable

✓ UNIT 7: Memory Allocation

---

20. Explain dynamic memory allocation using new and delete.

```
int *p=new int;
*p=100;
delete p;
```

---

21. Explain malloc and calloc with example.

```
int *p=(int*)malloc(sizeof(int));
free(p);
```

✅ UNIT 8: File Handling & Preprocessor

---

22. Write a program to write and read file in C++.

```
#include <fstream>
using namespace std;

int main(){
    ofstream out("file.txt");
    out<<"Hello";
    out.close();

    ifstream in("file.txt");
    string s;
    in>>s;
    cout<<s;
}
```

---

23. Explain preprocessor directives with examples.

```
#define PI 3.14
#ifdef PI
cout<<PI;
#endif
```

✅ UNIT 9: Classes

---

24. Explain class, constructor and copy constructor with example.

```
class Student{
    int roll;
public:
    Student(int r){ roll=r; }
    Student(const Student &s){
        roll=s.roll;
    }
};
```

✅ UNIT 10 & 11: Overloading, Inheritance, Polymorphism, Exception

---

25. Explain inheritance, polymorphism and exception handling with example.

```
class Base{
public:
    virtual void show(){
        cout<<"Base";
    }
};

class Derived: public Base{
public:
    void show(){
        cout<<"Derived";
    }
};

int main(){
    Base *ptr;
    Derived obj;
    ptr=&obj;
    ptr->show();  // Runtime polymorphism
}
```

Exception Example:

```
try{
    throw 10;
}
catch(int x){
    cout<<"Exception caught";
}
```

25 Very Long (15-Mark Level) Descriptive Questions with Detailed Answers (Including Programming Examples)
Covering **All 11 Units** of *Programming Fundamentals Using C/C++*.

✓ 1. Discuss the evolution of C and C++ in detail. Explain their features, advantages, and applications.

Answer:
C was developed by Dennis Ritchie in 1972 at Bell Labs for system programming and operating system development. It provided low-level memory access and high efficiency.

C++ was developed by Bjarne Stroustrup in 1985 as an extension of C. It introduced Object-Oriented Programming (OOP) features.

Features of C:

- Procedural programming
- Portable
- Efficient
- Structured programming

Features of C++:

- Supports OOP
- Data abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Templates
- Exception handling

Applications:

- Operating systems
- Embedded systems
- Game development
- Application software

✓ 2. Explain the complete structure of a C++ program and demonstrate compilation and execution process.

Answer:

Structure:

1. Documentation section
2. Preprocessor directives
3. Global declarations
4. main() function
5. Subprograms

```
#include <iostream>
using namespace std;

int main() {
    cout << "Program Structure Example";
    return 0;
}
```

Compilation Steps:

1. Write program (.cpp)
2. Compile (g++ program.cpp)
3. Linking
4. Execution (./a.out)

✓ 3. Explain data types, type modifiers and type casting in detail with suitable programs.

```
#include <iostream>
using namespace std;

int main() {
    int a = 10;
    float b = 5.5;
    double c = 12.34567;
    char d = 'A';

    float result = (float)a + b;  // Type casting
    cout << result;
}
```

Type Modifiers: short, long, signed, unsigned
Type casting: Implicit and Explicit

✓ 4. Discuss scope, storage classes, and lifetime of variables.

Storage Classes:

- auto
- static
- extern

- register

```
#include <iostream>
using namespace std;

void test() {
    static int x = 0;
    x++;
    cout << x << endl;
}

int main() {
    test();
    test();
}
```

Static variable retains value between function calls.

✅ 5. Explain all types of operators in C++ with examples.

Arithmetic, Logical, Relational, Bitwise, Assignment, Ternary.

```
int a=5, b=3;
cout << (a>b ? a:b);
```

✅ 6. Explain control statements in detail (if, switch, loops) with programs.

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;

    if(n>0)
        cout<<"Positive";
    else
        cout<<"Negative";
}
```

Loops: for, while, do-while.

✅ 7. Write a program to generate Fibonacci series using loop and recursion.

```
// Loop
for(int i=0;i<n;i++){
    cout<<a<<" ";
    int c=a+b;
    a=b;
    b=c;
}

// Recursion
int fib(int n){
    if(n<=1) return n;
    return fib(n-1)+fib(n-2);
}
```

✅ 8. Explain functions in detail including call by value and reference.

```
void swap(int &a,int &b){
    int t=a;
    a=b;
    b=t;
}
```

Call by reference modifies original variables.

✅ 9. Explain recursion with advantages and disadvantages.

Advantages:

- Simplifies code
- Suitable for tree problems

Disadvantages:

- More memory
- Stack overflow risk

✅ 10. Explain one-dimensional, two-dimensional and multidimensional arrays.

```
int arr[2][2]={{1,2},{3,4}};
```

Multidimensional: int arr[2][2][2];

✅ 11. Write a complete program for matrix multiplication.

```
#include <iostream>
using namespace std;

int main() {
    int a[2][2]={{1,2},{3,4}};
    int b[2][2]={{5,6},{7,8}};
    int c[2][2];

    for(int i=0;i<2;i++)
        for(int j=0;j<2;j++){
            c[i][j]=0;
            for(int k=0;k<2;k++)
                c[i][j]+=a[i][k]*b[k][j];
        }
}
```

✅ 12. Explain structures in detail with nested structures and arrays of structures.

```
struct Address{
    string city;
};

struct Student{
    int roll;
```

```
    Address addr;
};
```

✅ 13. Compare structure and union with memory diagram explanation.

Structure → separate memory
Union → shared memory

Union saves memory.

✅ 14. Explain pointers, pointer arithmetic, and pointer to pointer.

```
int x=10;
int *p=&x;
int **pp=&p;
```

✅ 15. Discuss passing arrays and pointers to functions.

Arrays passed as base address.

```
void display(int arr[], int n){}
```

✅ 16. Explain references and difference between pointers and references.

Reference:

```
int x=10;
int &r=x;
```

Difference:

- Pointer can change
- Reference cannot be null

✅ 17. Explain static and dynamic memory allocation with examples.

```
int *p=new int;
delete p;
```

Heap vs Stack memory.

✅ 18. Explain malloc, calloc, free vs new, delete.

malloc → C function
new → C++ operator
new calls constructor.

✅ 19. Explain file handling in C++ with read/write example.

```
ofstream out("file.txt");
out<<"Hello";
out.close();

ifstream in("file.txt");
string s;
in>>s;
```

✅ 20. Explain random file access with seekg() and seekp().

```
file.seekg(0, ios::beg);
```

✅ 21. Explain preprocessor directives and conditional compilation.

```
#define PI 3.14
#ifdef PI
cout<<PI;
#endif
```

✅ 22. Explain class, object, constructor, destructor with example.

```
class Test{
public:
    Test(){ cout<<"Constructor"; }
    ~Test(){ cout<<"Destructor"; }
};
```

✅ 23. Explain function overloading and operator overloading.

```
int add(int a,int b);
float add(float a,float b);
```

Operator overloading:

```
Complex operator+(Complex c);
```

✅ 24. Explain inheritance (single, multilevel, multiple) with program.

```
class A{};
class B: public A{};
class C: public B{};
```

✅ 25. Explain polymorphism and exception handling with example.

```
class Base{
public:
    virtual void show(){}
};

try{
    throw 5;
}
catch(int x){
    cout<<"Exception";
}
```

Polymorphism achieved through virtual functions.