620 MCQ for 18 Sessions

Session 1

🟢 EASY MCQs (1–10)

1. Which of the following is the correct extension of a C++ source file?

A. `.c`
B. `.cpp`
C. `.cp`
D. `.cxx`

✅ **Answer:** B
(Most commonly used extension)

---

2. Who developed C++?

A. Dennis Ritchie
B. James Gosling
C. Bjarne Stroustrup
D. Guido van Rossum

✅ **Answer:** C

---

3. C++ was initially called:

A. C Plus
B. Better C
C. C with Classes
D. Object C

✅ **Answer:** C

---

4. Which header file is required for `cout`?

A. `<stdio.h>`
B. `<conio.h>`
C. `<iostream>`
D. `<stdlib.h>`

✅ **Answer:** C

---

5. Which symbol is used for single-line comments in C++?

A. `/* */`
B. `//`
C. `#`
D. `--`

✅ **Answer:** B

---

6. What is the output of the following program?

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World";
    return 0;
}
```

A. Hello
B. World
C. HelloWorld
D. Hello World

✅ **Answer:** D

---

7. Which operator is used to insert data into `cout`?

A. `>>`
B. `<<`
C. `->`
D. `:`

✅ **Answer:** B

---

8. Which data type is used to store decimal values?

A. int
B. char
C. float
D. bool

✅ **Answer:** C

---

9. Which function is the entry point of a C++ program?

A. start()
B. begin()
C. main()
D. init()

✅ **Answer:** C

---

10. Which of the following is NOT a feature of C++?

A. Object-oriented
B. Platform independent
C. Low-level memory access
D. Procedural programming

✅ **Answer:** B
(C++ is platform dependent)

🟡 MEDIUM MCQs (11–20)

11. Which feature allows C++ to support both procedural and OOP?

A. Polymorphism
B. Abstraction
C. Multi-paradigm support
D. Encapsulation

✅ **Answer:** C

---

12. Difference between C and C++ is:

A. C supports classes, C++ does not
B. C++ supports OOP, C does not
C. C++ is older than C
D. C supports namespaces

✅ **Answer:** B

---

13. What will be the output?

```
#include <iostream>
using namespace std;
int main() {
    int a = 5, b = 10;
    cout << a + b;
}
```

A. 5
B. 10
C. 15
D. Error

✅ **Answer:** C

---

14. Which keyword is used to define a constant in C++?

A. constant
B. final
C. const
D. static

✅ **Answer:** C

---

15. What is the output?

```
int a = 5;
```

```
cout << a++;
```

A. 6
B. 5
C. Error
D. Undefined

✅ **Answer:** B
(Post-increment)

---

16. Which header is needed for mathematical functions like `pow()`?

A. `<math>`
B. `<cmath>`
C. `<stdlib>`
D. `<algorithm>`

✅ **Answer:** B

---

17. What is the output?

```
cout << sizeof(char);
```

A. 2
B. 4
C. 1
D. Depends on compiler

✅ **Answer:** C

---

18. Which operator is used for compound interest calculation?

A. `*`
B. `^`
C. `pow()`
D. `**`

✅ **Answer:** C
(C++ has no ^ power operator)

---

19. Which statement correctly swaps two numbers?

A. `a = b; b = a;`
B. `a = a + b; b = a - b; a = a - b;`
C. `swap(a, b, c);`
D. `swap(a, b, c, d);`

✅ **Answer:** B

---

20. Which is the correct formula for compound interest?

A. `P + RT`
B. `P(1 + R)^T`
C. `P * R * T`
D. `P - R - T`

✅ **Answer:** B

🔴 HARD & TRICKY MCQs (21–30)

21. What is the output?

```
int a = 2;
cout << a * a++ ;
```

A. 4
B. 6
C. 8
D. Undefined

✅ **Answer:** B
(2 × 3 after post-increment)

---

22. Which of the following is TRUE?

A. C++ does not support pointers
B. C++ supports both high-level and low-level programming
C. C++ cannot access memory directly
D. C++ does not support function overloading

✅ **Answer:** B

---

23. Output?

```
char ch = 'A';
cout << ch + 1;
```

A. B
B. A
C. 66
D. Error

✅ **Answer:** C
(ASCII: A = 65 → 65 + 1)

---

24. Which of these is NOT a valid identifier?

A. _value
B. value1
C. 1value
D. Value

✅ **Answer:** C

---

25. What happens if `return 0;` is omitted in `main()`?

A. Compilation error
B. Runtime error
C. Program crashes
D. Program runs normally

✅ **Answer:** D
(In modern C++, return 0 is implicit)

---

26. Output?

```
int a = 10, b = 20;
cout << (a > b ? a : b);
```

A. 10
B. 20
C. true
D. false

✅ **Answer:** B

---

27. What is the output?

```
cout << 5 / 2;
```

A. 2.5
B. 2
C. 3
D. Error

✅ **Answer:** B
(Integer division)

---

28. Which feature of C++ helps in code reuse?

A. Polymorphism
B. Inheritance
C. Encapsulation
D. Abstraction

✅ **Answer:** B

---

29. Which one is faster?

A. C
B. C++
C. Both same
D. Depends on compiler

✅ **Answer:** D
(Tricky exam answer)

---

30. What is the output?

```
int a = 5;
int &r = a;
r = 10;
cout << a;
```

A. 5
B. 10
C. Garbage
D. Error

✅ **Answer:** B
(Reference variable)

Session 2 & 3

🟢 EASY MCQs (15 Questions)

(Basics, definitions, simple snippets — confidence builders)

---

1. Which part of a C++ program contains `#include` statements?

A. Documentation section
B. Link section
C. Definition section
D. Global section

✅ **Answer:** B

---

2. Which of the following is a valid C++ token?

A. `@count`
B. `123abc`
C. `total_marks`
D. `#include`

✅ **Answer:** C

---

3. Tokens in C++ include all EXCEPT

A. Keywords
B. Identifiers
C. Constants
D. Comments

✅ **Answer:** D

---

4. Which keyword is used to declare a constant data member?

A. static
B. final
C. const
D. constant

✅ **Answer:** C

---

5. Which operator is used to assign a value?

A. `==`
B. `=`
C. `!=`
D. `<=`

✅ **Answer:** B

---

6. What is the output?

```
int a = 10;
```

```
cout << a;
```

A. Garbage
B. 0
C. 10
D. Error

✅ **Answer:** C

---

7. Which operator is used for logical AND?

A. `&`
B. `&&`
C. `|`
D. `||`

✅ **Answer:** B

---

8. Which of the following is a unary operator?

A. `+`
B. `*`
C. `++`
D. `==`

✅ **Answer:** C

---

9. Which operator is known as conditional operator?

A. `:`
B. `? :`
C. `if`
D. `switch`

✅ **Answer:** B

---

10. Static data members are

A. Created per object
B. Created once per class
C. Created per function
D. Created at runtime

✅ **Answer:** B

---

11. Default initialization of an int variable is

A. 0
B. Garbage
C. NULL
D. Depends on compiler

✅ **Answer:** B

---

12. Which operator checks equality?

A. `=`
B. `!=`
C. `==`
D. `>=`

✅ **Answer:** C

---

13. Which C++17 feature improves performance by avoiding copies?

A. Lambda
B. Move semantics
C. References
D. Pointers

✅ **Answer:** B

---

14. Which operator has the highest precedence?

A. `+`
B. `*`

C. `++`
D. `=`

✅ **Answer:** C

---

15. Which is a valid identifier?

A. `roll-no`
B. `1student`
C. `_student`
D. `class`

✅ **Answer:** C

🟡 MEDIUM MCQs (15 Questions)

(Concept + operator + class + snippet logic)

---

16. Which of the following must be initialized using constructor initializer list?

A. static members
B. const data members
C. local variables
D. global variables

✅ **Answer:** B

---

17. Output?

```
int a = 5, b = 10;
cout << (a > b);
```

A. true
B. false
C. 1
D. 0

✅ **Answer:** D
(false → 0)

---

18. Which operator cannot be overloaded?

A. `+`
B. `=`
C. `?:`
D. `[]`

✅ **Answer:** C

---

19. Static members are accessed using

A. Object name
B. Function name
C. Class name
D. Namespace

✅ **Answer:** C

---

20. Output?

```
int a = 5;
cout << a++ + ++a;
```

A. 10
B. 11
C. 12
D. Undefined

✅ **Answer:** B
(5 + 6)

---

21. Which operator is right associative?

A. `+`
B. `*`

C. `=`
D. `<`

✅ **Answer:** C

---

22. Which C++17 feature allows multiple initializations in if statement?

A. Lambda
B. Structured binding
C. if-init statement
D. constexpr

✅ **Answer:** C

---

23. Output?

```
bool x = true;
bool y = false;
cout << (x && y || x);
```

A. 0
B. 1
C. true
D. false

✅ **Answer:** B

---

24. Which operator is used for object creation?

A. `malloc`
B. `new`
C. `create`
D. `alloc`

✅ **Answer:** B

---

25. Constant members

A. Can be modified
B. Must be initialized
C. Are static by default
D. Are optional

✅ **Answer:** B

---

26. Output?

```
int a = 10;
cout << ~a;
```

A. 10
B. -10
C. -11
D. 11

✅ **Answer:** C
(Bitwise NOT)

---

27. Which operator checks logical OR?

A. `|`
B. `||`
C. `!`
D. `^`

✅ **Answer:** B

---

28. Static data members

A. Cannot be private
B. Cannot be accessed outside
C. Belong to class, not object
D. Belong to object

✅ **Answer:** C

---

29. Which operator has lowest precedence?

A. `=`
B. `*`
C. `+`
D. `++`

✅ **Answer:** A

---

30. Output?

```
int x = 0;
cout << (x ? 100 : 200);
```

A. 0
B. 100
C. 200
D. Error

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (15 Questions)

(CDAC-level traps, order of evaluation, static/const confusion)

---

31. Output?

```
int a = 3;
cout << a * a++ ;
```

A. 9
B. 12
C. 6
D. Undefined

✅ **Answer:** C

---

32. Which is TRUE about static members?

A. Initialized in constructor
B. Initialized outside class
C. Cannot be private
D. Object specific

✅ **Answer:** B

---

33. Output?

```
int a = 5;
cout << ++a + a++;
```

A. 11
B. 12
C. 13
D. Undefined

✅ **Answer:** B
(6 + 6)

---

34. Which token category does 42 belong to?

A. Identifier
B. Keyword
C. Constant
D. Operator

✅ **Answer:** C

---

35. Output?

```
int a = 10;
int b = 20;
cout << a < b << a > b;
```

A. 10
B. 01
C. 11
D. Error

✅ **Answer:** B
(true false → 1 0)

---

36. Which operator has highest precedence?

A. `()`
B. `*`
C. `++`
D. `=`

✅ **Answer:** A

---

37. Output?

```
int x = 5;
cout << x++ << ++x;
```

A. 56
B. 57
C. 67
D. Undefined

✅ **Answer:** B
(5 then 7)

---

38. Which operator is overloaded for streams?

A. `>>`
B. `<<`
C. Both
D. None

✅ **Answer:** C

---

39. Which C++17 feature allows grouping return values?

A. Lambda
B. Structured bindings

C. constexpr
D. namespace

✅ **Answer:** B

---

40. Output?

```
int a = 1;
cout << a++ + ++a + a;
```

A. 6
B. 7
C. 8
D. Undefined

✅ **Answer:** B
(1 + 3 + 3)

---

41. Static members are destroyed

A. When object is destroyed
B. At program end
C. When function ends
D. Never

✅ **Answer:** B

---

42. Output?

```
int a = 5;
cout << (a = 10);
```

A. 5
B. 10
C. true
D. Error

✅ **Answer:** B

---

43. Which operator is NOT associative?

A. `+`
B. `*`
C. `=`
D. `&&`

✅ **Answer:** C

---

44. Which expression is evaluated first?

A. Left to right always
B. Based on precedence
C. Based on associativity
D. Random

✅ **Answer:** B

---

45. In Student sorting (roll, DOB, marks), which concept is best?

A. Operator overloading
B. Function overloading
C. Comparator functions
D. Macros

✅ **Answer:** C

Session 4

🟢 EASY MCQs (1–20)

(Syntax, basics, direct outputs)

---

1. Which statement is used to execute code when a condition is true?

A. for
B. while
C. if
D. switch

✅ **Answer:** C

---

2. Which loop is guaranteed to execute at least once?

A. for
B. while
C. do-while
D. switch

✅ **Answer:** C

---

3. Which keyword exits a loop immediately?

A. continue
B. exit
C. break
D. stop

✅ **Answer:** C

---

4. Which keyword skips the current iteration?

A. return
B. break
C. continue
D. goto

✅ **Answer:** C

---

5. Which statement is used to select among multiple choices?

A. if
B. for
C. switch
D. while

✅ **Answer:** C

---

6. Output?

```
int x = 10;
if(x > 5)
  cout << "Yes";
else
  cout << "No";
```

A. Yes
B. No
C. Error
D. Nothing

✅ **Answer:** A

---

7. Array index in C++ starts from

A. 1
B. 0
C. -1
D. Depends on compiler

✅ **Answer:** B

---

8. Correct way to declare array of 5 integers

A. int arr;
B. int arr(5);
C. int arr[5];
D. int arr{5};

✅ **Answer:** C

---

9. Which loop is best when number of iterations is known?

A. while
B. do-while

C. for
D. switch

✅ **Answer:** C

---

10. Output?

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

A. 123
B. 012
C. 321
D. Error

✅ **Answer:** A

---

11. Which statement ends program execution?

A. break
B. continue
C. return
D. goto

✅ **Answer:** C

---

12. Which is a valid switch expression type?

A. float
B. double
C. int
D. string

✅ **Answer:** C

---

13. Default case in switch executes when

A. First case matches
B. No case matches

C. Always
D. Last case matches

✅ **Answer:** B

---

14. Output?

```
int i = 0;
while(i < 3){
  cout << i;
  i++;
}
```

A. 012
B. 123
C. 321
D. Infinite loop

✅ **Answer:** A

---

15. How many elements in array `int a[10];`?

A. 9
B. 10
C. 11
D. Depends

✅ **Answer:** B

---

16. Which statement skips remaining code in loop iteration?

A. break
B. return
C. continue
D. goto

✅ **Answer:** C

---

17. Output?

```
int x = 5;
if(x)
  cout<<"True";
```

A. False
B. True
C. Error
D. Nothing

✅ **Answer:** B

---

18. Which array stores rows and columns?

A. 1-D
B. 2-D
C. Multi-array
D. Jagged array

✅ **Answer:** B

---

19. Command line arguments are accessed using

A. cin
B. argv
C. argc
D. main

✅ **Answer:** B

---

20. argc represents

A. Argument values
B. Argument count
C. Argument size
D. Argument index

✅ **Answer:** B

🟡 MEDIUM MCQs (21–40)

(Logic, nested loops, switch, arrays, argc/argv)

---

21. Output?

```
int i=1;
do{
 cout<<i;
 i++;
}while(i<=3);
```

A. 123
B. 012
C. 13
D. Error

✅ **Answer:** A

---

22. Output?

```
for(int i=0;i<5;i++){
 if(i==3) break;
 cout<<i;
}
```

A. 0123
B. 012
C. 01234
D. 03

✅ **Answer:** B

---

23. Output?

```
for(int i=1;i<=5;i++){
 if(i==3) continue;
 cout<<i;
}
```

A. 12345
B. 1245
C. 1345
D. 12

✅ **Answer:** B

---

24. Switch case must end with

A. continue
B. break
C. return
D. goto

✅ **Answer:** B

---

25. Output?

```
int a[3]={10,20,30};
cout<<a[1];
```

A. 10
B. 20
C. 30
D. Error

✅ **Answer:** B

---

26. Output?

```
int arr[]={1,2,3,4};
cout<<sizeof(arr)/sizeof(arr[0]);
```

A. 3
B. 4
C. 8
D. Error

✅ **Answer:** B

---

27. Which loop checks condition before execution?

A. do-while
B. while
C. both
D. none

✅ **Answer:** B

---

28. Output?

```
int x=2;
switch(x){
 case 1: cout<<"One";
 case 2: cout<<"Two";
 case 3: cout<<"Three";
}
```

A. Two
B. TwoThree
C. OneTwoThree
D. Error

✅ **Answer:** B *(no break)*

---

29. Which keyword exits function immediately?

A. break
B. continue
C. return
D. exit

✅ **Answer:** C

---

30. Output?

```
int i=0;
```

```
for(;i<3;)
 cout<<i++;
```

A. 012
 B. 123
 C. 321
 D. Infinite

✅ **Answer:** A

---

31. 2D array declaration

A. int a(2,3);
 B. int a[2][3];
 C. int a[2,3];
 D. int a[][];

✅ **Answer:** B

---

32. Output?

```
int a[2][2]={{1,2},{3,4}};
cout<<a[1][0];
```

A. 1
 B. 2
 C. 3
 D. 4

✅ **Answer:** C

---

33. argc includes

A. Only arguments
 B. Only program name
 C. Program name + arguments
 D. Arguments only

✅ **Answer:** C

---

34. Output?

```
int x=0;
if(x==0)
 cout<<"Zero";
else
 cout<<"Non-zero";
```

A. Zero
B. Non-zero
C. Error
D. Nothing

✅ **Answer:** A

---

35. Nested loop means

A. Loop without condition
B. Loop inside loop
C. Infinite loop
D. Parallel loop

✅ **Answer:** B

---

36. Output?

```
for(int i=1;i<=2;i++)
 for(int j=1;j<=2;j++)
  cout<<i<<j<<" ";
```

A. 11 12 21 22
B. 11 21 12 22
C. 12 21
D. Error

✅ **Answer:** A

---

37. Which statement transfers control to calling function?

A. break
B. continue
C. return
D. goto

✅ **Answer:** C

---

38. Output?

```
int a[]={1,2,3};
cout<<a[3];
```

A. 3
B. 0
C. Garbage
D. Error

✅ **Answer:** C *(out of bounds)*

---

39. Switch case labels must be

A. variables
B. expressions
C. constants
D. strings

✅ **Answer:** C

---

40. Which loop is best for menu-driven programs?

A. for
B. while
C. do-while
D. switch

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (41–65)

(CDAC-level traps, evaluation order, arrays + CLI)

---

41. Output?

```
int i=0;
while(i<3)
 cout<<i++;
```

A. 012
B. 123
C. Infinite
D. Error

✅ **Answer:** A

---

42. Output?

```
int i=1;
for(;;){
 cout<<i++;
 if(i>3) break;
}
```

A. 123
B. 12
C. Infinite
D. Error

✅ **Answer:** A

---

43. Output?

```
int x=5;
if(x=0)
 cout<<"Yes";
else
 cout<<"No";
```

A. Yes
B. No
C. Error
D. Undefined

✅ **Answer:** B *(assignment)*

---

44. Output?

```
int a[5]={1,2};
cout<<a[3];
```

A. 0
B. 2
C. Garbage
D. Error

✅ **Answer:** A *(remaining initialized to 0)*

---

45. Output?

```
int i=0;
do{
 cout<<i;
}while(i++<0);
```

A. 0
B. 01
C. Infinite
D. Error

✅ **Answer:** A

---

46. What happens if break is missing in switch?

A. Error
B. Infinite
C. Fall-through
D. Skip default

✅ **Answer:** C

---

47. Output?

```
for(int i=0;i<5;i++);
 cout<<i;
```

A. 5
B. 01234
C. Error
D. Garbage

✅ **Answer:** A *(semicolon trap)*

---

48. Which loop can be infinite intentionally?

A. for(;;)
B. while(true)
C. do-while(true)
D. All

✅ **Answer:** D

---

49. Output?

```
int x=10;
switch(x){
 case 10: cout<<"Ten"; break;
 default: cout<<"Other";
}
```

A. Ten
B. Other
C. Error
D. Both

✅ **Answer:** A

---

50. Command line arguments are passed to

A. main()
B. constructor
C. global scope
D. namespace

✅ **Answer:** A

---

51. main() signature with arguments

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

argc means?
A. Argument values
B. Argument count
C. Argument size
D. Argument index

✅ **Answer:** B

---

52. Output?

```
int a[]={10,20,30};
for(int i: a)
 cout<<i;
```

A. 102030
B. 123
C. Error
D. Garbage

✅ **Answer:** A *(range-based loop)*

---

53. Which loop is entry-controlled?

A. do-while
B. while
C. for
D. B and C

✅ **Answer:** D

---

54. Output?

```
int i=0;
while(i++<3)
 cout<<i;
```

A. 123
B. 012
C. 234
D. Error

✅ **Answer:** A

---

55. Which statement is NOT a jump statement?

A. break
B. continue
C. return
D. switch

✅ **Answer:** D

---

56. Accessing array beyond size results in

A. Compilation error
B. Runtime error
C. Garbage value
D. Warning only

✅ **Answer:** C

---

57. Output?

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

A. 1
B. 2
C. 3
D. Error

✅ **Answer:** B

---

58. Which loop is preferred when iterations unknown?

A. for
B. while
C. do-while
D. switch

✅ **Answer:** B

---

59. Output?

```
int x=1;
switch(x){
 case 1:
 case 2: cout<<"Yes"; break;
 default: cout<<"No";
}
```

A. Yes
B. No
C. Error
D. Both

✅ **Answer:** A

---

60. Command line argument argv[0] contains

A. First argument
B. Program name
C. Last argument
D. Garbage

✅ **Answer:** B

---

61. Output?

```
int a[]={};
cout<<sizeof(a);
```

A. 0
B. Error
C. Garbage
D. Undefined

✅ **Answer:** B *(illegal)*

---

62. Which is safest loop exit?

A. break
B. goto
C. return
D. continue

✅ **Answer:** A

---

63. Output?

```
int i=5;
while(i--)
 cout<<i;
```

A. 43210
B. 54321
C. 4321
D. Error

✅ **Answer:** A

---

64. Which loop executes body at least once?

A. for
B. while

C. do-while
D. switch

✅ **Answer:** C

---

65. Best structure for menu-driven program

A. for + if
B. while + switch
C. if only
D. recursion

✅ **Answer:** B

Session 5

🟢 EASY MCQs (1–15)

---

1. What is a function in C++?

A. A variable
B. A block of reusable code
C. A class
D. A loop

✅ **Answer:** B

---

2. Which part of a function specifies return type and parameters?

A. Function body
B. Function call
C. Function prototype
D. Header file

✅ **Answer:** C

---

3. Which keyword is used to define an inline function?

A. static
B. inline
C. const
D. register

✅ **Answer:** B

---

4. Which header file contains math functions like `sqrt()`?

A. `<math>`
B. `<cmath>`
C. `<stdlib>`
D. `<algorithm>`

✅ **Answer:** B

---

5. Call by reference means

A. Passing value copy
B. Passing address
C. Passing constant
D. Passing object

✅ **Answer:** B

---

6. Output?

```
void fun() {
  cout << "Hello";
}
int main() {
  fun();
}
```

A. Hello
B. fun

C. Error
D. Nothing

✅ **Answer:** A

---

7. Which symbol is used for reference variables?

A. `*`
B. `&`
C. `#`
D. `@`

✅ **Answer:** B

---

8. Output?

```
int add(int a, int b) {
  return a + b;
}
int main() {
  cout << add(2,3);
}
```

A. 2
B. 3
C. 5
D. Error

✅ **Answer:** C

---

9. Inline functions are expanded

A. At runtime
B. At compile time
C. At linking
D. Never

✅ **Answer:** B

---

10. Which is a math library function?

A. pow()
B. print()
C. scan()
D. display()

✅ **Answer:** A

---

11. Function prototype is required when

A. Function is before main
B. Function is after main
C. Inline function
D. Recursive function

✅ **Answer:** B

---

12. Output?

```
int square(int x) {
  return x * x;
}
cout << square(4);
```

A. 8
B. 12
C. 16
D. Error

✅ **Answer:** C

---

13. Which is correct function prototype?

A. `int add(a,b);`
B. `add(int,int);`
C. `int add(int,int);`
D. `int add;`

✅ **Answer:** C

14. Which function does NOT return a value?

A. int
B. float
C. void
D. double

✅ **Answer:** C

15. Inline functions are best suited for

A. Large functions
B. Recursive functions
C. Small frequently used functions
D. File handling

✅ **Answer:** C

🟡 MEDIUM MCQs (16–30)

16. Output?

```
void change(int &x) {
  x = 10;
}
int main() {
  int a = 5;
  change(a);
  cout << a;
}
```

A. 5
B. 10
C. Garbage
D. Error

✅ **Answer:** B

---

17. Which type of function call allows modification of actual arguments?

A. Call by value
B. Call by name
C. Call by reference
D. Call by result

✅ **Answer:** C

---

18. Output?

```
inline int cube(int x) {
  return x * x * x;
}
int main() {
  cout << cube(3);
}
```

A. 6
B. 9
C. 27
D. Error

✅ **Answer:** C

---

19. Which function requires `<cmath>`?

A. strlen()
B. sqrt()
C. printf()
D. cin()

✅ **Answer:** B

---

20. Output?

```
int x = 5;
void test(int x) {
  x = 10;
}
int main() {
  test(x);
  cout << x;
}
```

A. 5
B. 10
C. Error
D. Garbage

✅ **Answer:** A *(call by value)*

---

21. Which function feature improves performance?

A. Prototyping
B. Recursion
C. Inline
D. Overloading

✅ **Answer:** C

---

22. Output?

```
int fun(int &x) {
  return x++;
}
int main() {
  int a = 5;
  cout << fun(a) << " " << a;
}
```

A. 5 6
B. 6 6
C. 6 5
D. 5 5

✅ **Answer:** A

---

23. Which is TRUE about inline functions?

A. Always inlined
B. Compiler decides inlining
C. Cannot have parameters
D. Cannot return value

✅ **Answer:** B

---

24. Output?

```
double x = sqrt(16);
cout << x;
```

A. 4
B. 4.0
C. Error
D. Garbage

✅ **Answer:** B

---

25. Which function passes address implicitly?

A. Call by value
B. Call by reference
C. Inline
D. Recursive

✅ **Answer:** B

---

26. Output?

```
int add(int a, int b=10) {
  return a + b;
}
int main() {
  cout << add(5);
}
```

A. 5
B. 10
C. 15
D. Error

✅ **Answer:** C

---

27. Inline functions increase

A. Execution time
B. Code size
C. Runtime memory
D. Stack usage

✅ **Answer:** B

---

28. Output?

```
int fun(int x) {
  return x * fun(x-1);
}
```

A. Recursive function
B. Infinite loop
C. Inline function
D. Error

✅ **Answer:** A *(definition only)*

---

29. Which math function returns power?

A. square()
B. pow()
C. sqrt()
D. abs()

✅ **Answer:** B

---

30. Which is NOT a form of function?

A. Inline
B. Recursive
C. Lambda
D. Virtual

✅ **Answer:** D *(virtual is member function concept)*

🔴 HARD & TRICKY MCQs (31–45)

---

31. Output?

```
int fun(int &x) {
  x += 5;
  return x;
}
int main() {
  int a = 5;
  cout << fun(a) << " " << a;
}
```

A. 5 5
B. 10 5
C. 10 10
D. Error

✅ **Answer:** C

---

32. Output?

```
inline int fun(int x) {
  return x++;
}
int main() {
  int a = 5;
  cout << fun(a) << " " << a;
}
```

A. 5 6
B. 6 5
C. 5 5
D. 6 6

✅ **Answer:** C *(inline ≠ reference)*

---

33. Which function cannot be inline?

A. Small function
B. Recursive function
C. Static function
D. Const function

✅ **Answer:** B

---

34. Output?

```
int fun(int x, int y) {
  return x + y;
}
int main() {
  cout << fun(5, 10.5);
}
```

A. 15
B. 15.5
C. Error
D. Garbage

✅ **Answer:** A *(implicit conversion)*

---

35. Which is TRUE about function prototype?

A. Defines logic
B. Allocates memory
C. Declares function signature
D. Executes function

✅ **Answer:** C

36. Output?

```
int x = 10;
int& ref = x;
ref = 20;
cout << x;
```

A. 10
B. 20
C. Garbage
D. Error

✅ **Answer:** B

37. Inline functions are replaced by

A. Function calls
B. Macros
C. Function body
D. Jump instructions

✅ **Answer:** C

38. Which math function returns absolute value?

A. abs()
B. fabs()
C. mod()
D. pow()

✅ **Answer:** A

39. Output?

```
int fun(int x) {
  static int y = 0;
  y += x;
  return y;
}
```

```
int main() {
  cout << fun(5) << " " << fun(5);
}
```

A. 5 5
B. 5 10
C. 10 10
D. Error

✅ **Answer:** B

---

40. Which function feature avoids function call overhead?

A. Recursion
B. Inline
C. Overloading
D. Prototyping

✅ **Answer:** B

---

41. Output?

```
int fun(int &x) {
  return ++x;
}
int main() {
  int a = 10;
  cout << fun(a);
}
```

A. 10
B. 11
C. 12
D. Error

✅ **Answer:** B

---

42. Which is safest way to modify actual parameters?

A. Global variables
B. Call by reference
C. Macros
D. Inline

✅ **Answer:** B

---

43. Inline functions are expanded

A. Always
B. Never
C. If compiler decides
D. Only once

✅ **Answer:** C

---

44. Output?

```
cout << pow(2,3);
```

A. 6
B. 8
C. 9
D. Error

✅ **Answer:** B

---

45. Best use case of inline functions

A. File handling
B. Recursion
C. Small getter/setter
D. Sorting

✅ **Answer:** C

Session 6 & 7

🟢 EASY MCQs (1–25)

---

1. What is a pointer?

A. Variable storing value
B. Variable storing address
C. Constant
D. Function

✅ **Answer:** B

---

2. Which operator gives address of a variable?

A. `*`
B. `&`
C. `->`
D. `.`

✅ **Answer:** B

---

3. Which operator is used to access value at address?

A. `&`
B. `*`
C. `.`
D. `::`

✅ **Answer:** B

---

4. Output?

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

A. Address
B. 10

C. Garbage
D. Error

✅ **Answer:** B

---

5. Default value of an uninitialized pointer is

A. 0
B. NULL
C. Garbage
D. Address of 0

✅ **Answer:** C

---

6. Which operator allocates memory dynamically?

A. malloc
B. calloc
C. new
D. alloc

✅ **Answer:** C

---

7. Output?

```
int *p = new int(5);
cout << *p;
```

A. Garbage
B. 0
C. 5
D. Error

✅ **Answer:** C

---

8. Which operator frees memory allocated by `new`?

A. free
B. delete

C. remove
D. clear

✅ **Answer:** B

---

9. Which pointer refers to current object?

A. self
B. this
C. current
D. object

✅ **Answer:** B

---

10. Output?

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

A. 1
B. 2
C. 3
D. Garbage

✅ **Answer:** B

---

11. What does typedef do?

A. Declares variable
B. Creates alias
C. Defines constant
D. Allocates memory

✅ **Answer:** B

---

12. Example of typedef

```
typedef int marks;
marks m = 10;
```

marks is a
A. Variable
B. Function
C. Alias
D. Pointer

✅ **Answer:** C

---

13. Which header is needed for malloc?

A. `<iostream>`
B. `<stdlib.h>`
C. `<new>`
D. `<memory>`

✅ **Answer:** B

---

14. Output?

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

A. 10
B. Address
C. 0
D. Error

✅ **Answer:** B

---

15. Enum is used to

A. Store string
B. Store floating values
C. Store named constants
D. Allocate memory

✅ **Answer:** C

---

16. Output?

```
enum Day {MON, TUE, WED};
cout << MON;
```

A. MON
B. 0
C. 1
D. Error

✅ **Answer:** B

---

17. Which is safer in C++?

A. malloc
B. calloc
C. new
D. realloc

✅ **Answer:** C

---

18. Pointer arithmetic increases by

A. 1 byte
B. Address + 1
C. Size of datatype
D. Random

✅ **Answer:** C

---

19. Output?

```
int a = 5;
int *p = &a;
*p = 10;
cout << a;
```

A. 5
B. 10
C. Garbage
D. Error

✅ **Answer:** B

---

20. Which pointer stores address of pointer?

A. Single pointer
B. Double pointer
C. Null pointer
D. Void pointer

✅ **Answer:** B

---

21. Output?

```
int a = 5;
int **p;
int *q = &a;
p = &q;
cout << **p;
```

A. 5
B. Address
C. Garbage
D. Error

✅ **Answer:** A

---

22. Which function frees memory allocated by malloc?

A. delete
B. free
C. clear
D. remove

✅ **Answer:** B

---

23. Which is invalid?

A. int *p;*
B. int p;
C. int * p;
D. int &p;

✅ **Answer:** D *(needs initialization)*

---

24. Output?

```
int arr[2] = {10,20};
int *p = arr;
cout << *p++;
```

A. 10
B. 20
C. Garbage
D. Error

✅ **Answer:** A

---

25. new operator returns

A. Value
B. Address
C. Reference
D. Index

✅ **Answer:** B

🟡 MEDIUM MCQs (26–50)

---

26. Difference between malloc and new

A. new calls constructor
B. malloc initializes memory

C. malloc returns object
D. new works in C only

✅ **Answer:** A

---

27. Output?

```
int *p = new int[3]{1,2,3};
cout << p[2];
```

A. 1
B. 2
C. 3
D. Garbage

✅ **Answer:** C

---

28. Memory allocated using new[] should be freed using

A. delete
B. delete[]
C. free
D. remove

✅ **Answer:** B

---

29. Output?

```
int a = 10;
int *p = &a;
int *q = p;
*q = 20;
cout << a;
```

A. 10
B. 20
C. Garbage
D. Error

✅ **Answer:** B

---

30. Which pointer points to class object?

A. this
B. Object pointer
C. Class pointer
D. Static pointer

✅ **Answer:** B

---

31. Output?

```
class A {
public:
 int x;
 void set(int x) {
  this->x = x;
 }
};
int main() {
 A obj;
 obj.set(10);
 cout << obj.x;
}
```

A. 0
B. 10
C. Garbage
D. Error

✅ **Answer:** B

---

32. Output?

```
char str[] = "ABC";
char *p = str;
while(*p)
 cout << *p++;
```

A. ABC
B. A
C. Garbage
D. Error

✅ **Answer:** A

---

33. Which pointer can point to any datatype?

A. int pointer
B. void pointer
C. char pointer
D. double pointer

✅ **Answer:** B

---

34. Output?

```
int a = 5;
void fun(int *p) {
 *p = 10;
}
int main() {
 fun(&a);
 cout << a;
}
```

A. 5
B. 10
C. Error
D. Garbage

✅ **Answer:** B

---

35. Which memory allocation initializes to zero?

A. malloc
B. new
C. calloc
D. realloc

✅ **Answer:** C

---

36. Output?

```
int a = 5;
int *p = &a;
cout << sizeof(p);
```

A. sizeof(int)
B. sizeof(address)
C. 4
D. Depends on architecture

✅ **Answer:** D

---

37. Which is TRUE?

A. delete can free malloc memory
B. free can free new memory
C. Mixing free & delete is unsafe
D. malloc calls constructor

✅ **Answer:** C

---

38. Output?

```
enum Color {RED=5, GREEN, BLUE};
cout << GREEN;
```

A. 0
B. 5
C. 6
D. 7

✅ **Answer:** C

---

39. Which operator accesses members via pointer?

A. `.`
B. `::`
C. `->`
D. `*`

✅ **Answer:** C

---

40. Output?

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

A. 1
B. 2
C. 3
D. Error

✅ **Answer:** C

---

41. Typedef is replaced at

A. Runtime
B. Compile time
C. Link time
D. Execution

✅ **Answer:** B

---

42. Output?

```
char a[] = "HELLO";
char b[10];
int i=0;
while(a[i]) {
 b[i] = a[i];
 i++;
}
b[i]='\0';
```

```
cout << b;
```

A. HELLO
B. H
C. Garbage
D. Error

✅ **Answer:** A *(string copy)*

---

43. Which is NOT dynamic memory function?

A. new
B. delete
C. malloc
D. sizeof

✅ **Answer:** D

---

44. Pointer arithmetic on void pointer is

A. Allowed
B. Not allowed
C. Compiler dependent
D. Always increments by 1

✅ **Answer:** B

---

45. Output?

```
int a = 10;
int *p = &a;
p = NULL;
cout << p;
```

A. 0
B. 10
C. Garbage
D. Error

✅ **Answer:** A

---

46. Which is safer for objects?

A. malloc
B. calloc
C. new
D. realloc

✅ **Answer:** C

---

47. Output?

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

A. 1
B. 2
C. 3
D. Error

✅ **Answer:** B

---

48. Which pointer holds no valid address?

A. Dangling pointer
B. Null pointer
C. Void pointer
D. Wild pointer

✅ **Answer:** B

---

49. realloc is used to

A. Allocate memory
B. Free memory
C. Resize memory
D. Copy memory

✅ **Answer:** C

---

50. this pointer is

A. Static
B. Constant pointer
C. Global
D. Local

✅ **Answer:** B *(cannot change address)*

🔴 HARD & TRICKY MCQs (51–75)

---

51. Output?

```
int a = 5;
int *p = &a;
cout << *p++ << " " << a;
```

A. 5 5
B. 5 garbage
C. Undefined
D. 6 5

✅ **Answer:** A

---

52. Output?

```
int *p = new int;
cout << *p;
```

A. 0
B. Garbage
C. Error
D. NULL

✅ **Answer:** B

53. Dangling pointer occurs when

A. Pointer not initialized
B. Memory freed but pointer used
C. Pointer is NULL
D. Pointer out of scope

✅ **Answer:** B

---

54. Output?

```
int a = 10;
int *p = &a;
delete p;
cout << *p;
```

A. 10
B. 0
C. Garbage
D. Error

✅ **Answer:** C *(dangling)*

---

55. Which string function implementation is correct?

```
int strlen(char *s) {
 int i=0;
 while(s[i]) i++;
 return i;
}
```

A. Correct
B. Missing '\0'
C. Wrong return
D. Error

✅ **Answer:** A

---

56. Output?

```
int *p = (int*)malloc(sizeof(int));
*p = 10;
cout << *p;
```

A. 10
B. Garbage
C. Error
D. NULL

✅ **Answer:** A

---

57. Which is NOT allowed?

A. delete ptr;
B. delete[] ptr;
C. free(ptr);
D. free(new int);

✅ **Answer:** D

---

58. Output?

```
char a[]="AB";
char b[]="CD";
int i=0;
while(a[i]) i++;
int j=0;
while(b[j]) a[i++]=b[j++];
a[i]='\0';
cout<<a;
```

A. AB
B. CD
C. ABCD
D. Error

✅ **Answer:** C *(string concatenate)*

---

59. Which pointer problem causes memory leak?

A. Dangling pointer
B. Wild pointer
C. Lost pointer
D. Null pointer

✅ **Answer:** C

---

60. Output?

```
int a = 5;
int *p = &a;
int **q = &p;
cout << **q;
```

A. 5
B. Address
C. Garbage
D. Error

✅ **Answer:** A

---

61. new vs malloc — TRUE

A. malloc calls constructor
B. new returns void*
C. new throws exception
D. malloc is type safe

✅ **Answer:** C

---

62. Output?

```
int a=5;
int *p=&a;
*p += 5;
cout << a;
```

A. 5
B. 10
C. Garbage
D. Error

✅ **Answer:** B

---

63. Which pointer error is most dangerous?

A. Null pointer
B. Dangling pointer
C. Void pointer
D. Double pointer

✅ **Answer:** B

---

64. Output?

```
int *p = new int[2];
p[0]=10; p[1]=20;
delete p;
cout << p[1];
```

A. 20
B. 0
C. Garbage
D. Error

✅ **Answer:** C *(wrong delete)*

---

65. Which avoids memory leak?

A. delete after new
B. free after malloc
C. delete[] after new[]
D. All

✅ **Answer:** D

---

66. this pointer points to

A. Calling object
B. Class
C. Function
D. Compiler

✅ **Answer:** A

---

67. Output?

```
int *p = nullptr;
cout << p;
```

A. 0
B. NULL
C. Garbage
D. Error

✅ **Answer:** A

---

68. Which is safest string copy without <string.h>?

A. strcpy
B. memcpy
C. Loop with '\0'
D. strcat

✅ **Answer:** C

---

69. realloc preserves

A. New data only
B. Old data
C. Zero values
D. Garbage

✅ **Answer:** B

---

70. Output?

```
int a=5;
int *p=&a;
int *&r=p;
*r=10;
cout<<a;
```

A. 5
B. 10
C. Garbage
D. Error

✅ **Answer:** B

---

71. Which pointer type is fastest?

A. int*
B. char*
C. Depends on use
D. void*

✅ **Answer:** C

---

72. sizeof(pointer) gives

A. Size of value
B. Size of address
C. Size of array
D. Value

✅ **Answer:** B

---

73. Output?

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

A. 3
B. 4

C. 5
D. 6

✅ **Answer:** C

---

74. Which pointer cannot be dereferenced?

A. Void pointer
B. Null pointer
C. Dangling pointer
D. All

✅ **Answer:** D

---

75. Best practice

A. Mix malloc & delete
B. Avoid delete
C. Match new/delete
D. Use global pointers

✅ **Answer:** C

Session 8

🟢 EASY MCQs (1–10)

---

1. Object-Oriented Programming is based on

A. Algorithms
B. Functions
C. Objects
D. Variables

✅ **Answer:** C

---

2. Which of the following is NOT an OOP concept?

A. Encapsulation
B. Inheritance
C. Compilation
D. Polymorphism

✅ **Answer:** C

---

3. A class is

A. An object
B. A blueprint of object
C. A variable
D. A function

✅ **Answer:** B

---

4. Which access specifier allows access everywhere?

A. private
B. protected
C. public
D. default

✅ **Answer:** C

---

5. Default access specifier in a class is

A. public
B. private
C. protected
D. global

✅ **Answer:** B

---

6. Output?

```
class Test {
public:
 int x = 10;
};
int main() {
 Test t;
 cout << t.x;
}
```

A. 0
B. 10
C. Garbage
D. Error

✅ **Answer:** B

---

7. Which keyword is used to create a namespace?

A. package
B. module
C. namespace
D. scope

✅ **Answer:** C

---

8. Namespace helps to

A. Speed up program
B. Avoid name conflicts
C. Create objects
D. Access private members

✅ **Answer:** B

---

9. Which operator is used to access class members using object?

A. `->`
B. `.`
C. `::`
D. `*`

✅ **Answer:** B

---

10. Object is created using

A. class name only
B. constructor only
C. class name + variable
D. new keyword mandatory

✅ **Answer:** C

🟡 MEDIUM MCQs (11–20)

---

11. Which OOP concept binds data and functions together?

A. Inheritance
B. Polymorphism
C. Encapsulation
D. Abstraction

✅ **Answer:** C

---

12. Output?

```
class A {
 int x = 5;
public:
 int getX() { return x; }
};
int main() {
 A a;
 cout << a.getX();
}
```

A. 5
B. Garbage
C. Error
D. 0

✅ **Answer:** A

---

13. Members declared as private are accessible

A. Everywhere
B. Outside class
C. Inside class only
D. In namespace

✅ **Answer:** C

---

14. Which access specifier allows access in derived class?

A. private
B. protected
C. public
D. all

✅ **Answer:** B

---

15. Output?

```
namespace N {
 int x = 10;
}
int main() {
 cout << N::x;
}
```

A. 0
B. 10
C. Error
D. Garbage

✅ **Answer:** B

---

16. Which symbol is used for scope resolution?

A. `.`
B. `->`
C. `::`
D. `:`

✅ **Answer:** C

---

17. Which statement is TRUE?

A. One class can have multiple objects
B. One object can have multiple classes
C. Namespace creates object
D. Class executes code

✅ **Answer:** A

---

18. Output?

```
class Student {
public:
 int roll;
};
int main() {
 Student s;
 s.roll = 101;
 cout << s.roll;
}
```

A. 0
B. 101
C. Garbage
D. Error

✅ **Answer:** B

---

19. Why use namespaces in large projects?

A. Reduce memory
B. Increase speed

C. Avoid naming collisions
D. Support inheritance

✅ **Answer:** C

---

20. Access specifier used for data hiding

A. public
B. protected
C. private
D. namespace

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
class A {
 int x = 10;
};
int main() {
 A a;
 cout << a.x;
}
```

A. 10
B. 0
C. Garbage
D. Compilation error

✅ **Answer:** D *(private by default)*

---

22. Output?

```
namespace A {
 int x = 5;
}
```

```
namespace B {
 int x = 10;
}
int main() {
 cout << A::x + B::x;
}
```

A. 5
B. 10
C. 15
D. Error

✅ **Answer:** C

---

23. Which concept is achieved using access specifiers?

A. Polymorphism
B. Abstraction
C. Encapsulation
D. Inheritance

✅ **Answer:** C

---

24. Output?

```
class Test {
public:
 static int x;
};
int Test::x = 10;
int main() {
 Test t1, t2;
 t1.x = 20;
 cout << t2.x;
}
```

A. 10
B. 20

C. Garbage
D. Error

✅ **Answer:** B

---

25. Which is NOT allowed?

A. Class inside namespace
B. Namespace inside class
C. Object inside namespace
D. Multiple namespaces

✅ **Answer:** B

---

26. Output?

```
namespace N {
 class A {
 public:
  int x = 5;
 };
}
int main() {
 N::A obj;
 cout << obj.x;
}
```

A. 5
B. Error
C. Garbage
D. 0

✅ **Answer:** A

---

27. Best way to sort students by roll, DOB, marks

A. Using global variables
B. Using namespaces
C. Using comparator functions
D. Using macros

✅ **Answer:** C

---

28. Which OOP concept allows same function name, different behavior?

A. Encapsulation
B. Inheritance
C. Polymorphism
D. Abstraction

✅ **Answer:** C

---

29. Output?

```
class A {
public:
 A() { cout << "A"; }
};
int main() {
 A obj;
}
```

A. A
B. Nothing
C. Error
D. Garbage

✅ **Answer:** A

---

30. Real-world example of class and object

A. int and variable
B. Function and call
C. Blueprint and house
D. Loop and condition

✅ **Answer:** C

Session 9

🟢 EASY MCQs (1–10)

---

1. What is a constructor?

A. A function to destroy object
B. A function called automatically when object is created
C. A static function
D. A virtual function

✅ **Answer:** B

---

2. Constructor name must be

A. Same as function name
B. Same as class name
C. Any valid identifier
D. Same as object name

✅ **Answer:** B

---

3. Does a constructor have a return type?

A. Yes
B. No
C. Only void
D. Only int

✅ **Answer:** B

---

4. Which constructor takes arguments?

A. Default constructor
B. Copy constructor
C. Parameterized constructor
D. Destructor

✅ **Answer:** C

---

5. Output?

```
class A {
public:
 A() { cout << "A"; }
};
int main() {
 A obj;
}
```

A. Nothing
B. A
C. Error
D. Garbage

✅ **Answer:** B

---

6. Destructor name starts with

A. `#`
B. `!`
C. ~
D. `@`

✅ **Answer:** C

---

7. How many destructors can a class have?

A. 0
B. 1
C. 2
D. Unlimited

✅ **Answer:** B

---

8. When is destructor called?

A. At object creation
B. At compile time
C. At object destruction
D. At function call

✅ **Answer:** C

---

9. Which operator is used for dynamic object creation?

A. malloc
B. calloc
C. new
D. create

✅ **Answer:** C

---

10. Copy constructor is used to

A. Copy file
B. Copy function
C. Copy object
D. Copy pointer

✅ **Answer:** C

🟡 MEDIUM MCQs (11–20)

---

11. Output?

```
class A {
public:
 A() { cout << "D"; }
 A(int x) { cout << "P"; }
};
int main() {
 A obj(10);
}
```

A. D
B. P
C. DP
D. Error

✅ **Answer:** B

---

12. Which is a valid copy constructor?

A. `A(A obj)`
B. `A(A &obj)`
C. `A(const A &obj)`
D. B and C

✅ **Answer:** D

---

13. Output?

```
class A {
public:
 A() { cout << "A"; }
 ~A() { cout << "B"; }
};
int main() {
 A obj;
 cout << "C";
}
```

A. ABC
B. ACB
C. CAB
D. BAC

✅ **Answer:** B
(Destructor runs last)

---

14. Dynamic initialization of object means

A. Static allocation
B. Runtime allocation
C. Compile-time allocation
D. Stack allocation

✅ **Answer:** B

---

15. Output?

```
class A {
public:
 A() { cout << "A"; }
};
int main() {
 A *p = new A();
}
```

A. A
B. Error
C. Garbage
D. Nothing

✅ **Answer:** A

---

16. What happens if destructor is not written explicitly?

A. Compilation error
B. Runtime error
C. Compiler provides default destructor
D. Program crashes

✅ **Answer:** C

---

17. Output?

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

```
int main() {
 A *p = new A();
 delete p;
}
```

A. AB
B. BA
C. A
D. B

✅ **Answer:** A

---

18. Which constructor is called when object is passed by value?

A. Default constructor
B. Parameterized constructor
C. Copy constructor
D. Destructor

✅ **Answer:** C

---

19. How many constructors can a class have?

A. 1
B. 2
C. Unlimited (with different parameters)
D. Depends on compiler

✅ **Answer:** C

---

20. Destructor is mainly used to

A. Allocate memory
B. Initialize object
C. Free resources
D. Copy object

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
class A {
public:
 A() { cout << "A"; }
 A(const A &) { cout << "C"; }
};
void fun(A obj) {}
int main() {
 A a;
 fun(a);
}
```

A. A
B. AC
C. AAC
D. Error

✅ **Answer:** B
(copy constructor called)

---

22. Output?

```
class A {
public:
 A() { cout << "A"; }
 ~A() { cout << "D"; }
};
int main() {
 {
  A obj;
 }
 cout << "X";
}
```

A. ADX
B. AXD

C. ADX
D. XAD

✅ **Answer:** A
(Destructor when block ends)

---

23. Which constructor is NOT inherited?

A. Default
B. Parameterized
C. Copy
D. All constructors

✅ **Answer:** D

---

24. Output?

```
class A {
 int x;
public:
 A(int a) : x(a) { cout << x; }
};
int main() {
 A obj = 10;
}
```

A. 0
B. 10
C. Error
D. Garbage

✅ **Answer:** B

---

25. Destructor is called in which order?

A. Same as constructor
B. Reverse of constructor
C. Random
D. Depends on compiler

✅ **Answer:** B

---

26. Output?

```
class A {
public:
 A() { cout << "A"; }
};
class B {
 A a;
public:
 B() { cout << "B"; }
};
int main() {
 B obj;
}
```

A. AB
B. BA
C. A
D. Error

✅ **Answer:** A
(Member object constructed first)

---

27. Output?

```
class A {
public:
 A() { cout << "A"; }
 ~A() { cout << "D"; }
};
int main() {
 A *p = new A();
}
```

A. AD
B. A

C. D
D. Error

✅ **Answer:** B
(delete not called → destructor not executed)

---

28. Which is TRUE about destructor?

A. Can take parameters
B. Can be overloaded
C. Cannot be static
D. Returns value

✅ **Answer:** C

---

29. Inner class in C++ means

A. Class inside function
B. Class inside class
C. Class inside namespace
D. Class inside main

✅ **Answer:** B

---

30. Output?

```
class Outer {
public:
 class Inner {
 public:
  Inner() { cout << "I"; }
 };
};
int main() {
 Outer::Inner obj;
}
```

A. I
B. O

C. Error
D. Nothing

✅ **Answer:** A

Session 10

🟢 EASY MCQs (1–10)

---

1. What is inheritance in C++?

A. Creating objects
B. Reusing existing class features
C. Overloading functions
D. Memory allocation

✅ **Answer:** B

---

2. Which class is inherited?

A. Child class
B. Derived class
C. Base class
D. Friend class

✅ **Answer:** C

---

3. Syntax of inheritance

```
class Derived : access Base
```

Access can be:
A. public
B. private
C. protected
D. All

✅ **Answer:** D

---

4. Which inheritance has one base and one derived class?

A. Multiple
B. Multilevel
C. Single
D. Hybrid

✅ **Answer:** C

---

5. Output?

```
class A {
public:
 void show() { cout << "A"; }
};
class B : public A {};
int main() {
 B obj;
 obj.show();
}
```

A. A
B. B
C. Error
D. Nothing

✅ **Answer:** A

---

6. Which inheritance allows multiple base classes?

A. Single
B. Multilevel
C. Multiple
D. Hierarchical

✅ **Answer:** C

---

7. Which inheritance forms a tree-like structure?

A. Single
B. Multiple
C. Hierarchical
D. Hybrid

✅ **Answer:** C

---

8. Which keyword prevents ambiguity in multiple inheritance?

A. friend
B. static
C. virtual
D. protected

✅ **Answer:** C

---

9. Derived class can access protected members of base class

A. Yes
B. No
C. Only public
D. Only private

✅ **Answer:** A

---

10. Inheritance supports

A. Code reuse
B. Encapsulation
C. Polymorphism
D. All

✅ **Answer:** D

🟡 MEDIUM MCQs (11–20)

---

11. Output?

```
class A {
public:
 A() { cout << "A"; }
};
class B : public A {
public:
 B() { cout << "B"; }
};
int main() {
 B obj;
}
```

A. AB
B. BA
C. A
D. B

✅ **Answer:** A
(Base constructor first)

---

12. Which inheritance creates ambiguity problem?

A. Single
B. Multilevel
C. Multiple
D. Hierarchical

✅ **Answer:** C

---

13. Output?

```
class A {
public:
 void show() { cout << "A"; }
};
class B : public A {
public:
 void show() { cout << "B"; }
};
int main() {
 B obj;
 obj.show();
}
```

A. A
B. B
C. AB
D. Error

✅ **Answer:** B *(function overriding)*

---

14. Which inheritance combines two or more types?

A. Multiple
B. Multilevel
C. Hybrid
D. Hierarchical

✅ **Answer:** C

---

15. Output?

```
class A {
protected:
 int x = 10;
};
class B : public A {
public:
 void show() { cout << x; }
};
int main() {
 B obj;
 obj.show();
}
```

A. 0
B. 10

C. Error
D. Garbage

✅ **Answer:** B

---

16. Constructor execution order

A. Derived → Base
B. Base → Derived
C. Random
D. Compiler dependent

✅ **Answer:** B

---

17. Which inheritance has base → derived → derived chain?

A. Multiple
B. Multilevel
C. Hierarchical
D. Hybrid

✅ **Answer:** B

---

18. Output?

```
class A {
public:
 int x = 5;
};
class B : private A {
public:
 void show() { cout << x; }
};
int main() {
 B obj;
 obj.show();
}
```

A. 5
B. Error
C. Garbage
D. 0

✅ **Answer:** A

---

19. Which members are NOT inherited?

A. Public
B. Protected
C. Private
D. All inherited

✅ **Answer:** C

---

20. Which feature helps avoid duplicate base class copies?

A. Friend
B. Namespace
C. Virtual base class
D. Protected

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
class A {
public:
 A() { cout << "A"; }
};
class B : virtual public A {
public:
 B() { cout << "B"; }
};
class C : virtual public A {
public:
 C() { cout << "C"; }
};
class D : public B, public C {
public:
 D() { cout << "D"; }
};
int main() {
 D obj;
}
```

A. ABCD
 B. A B C D
 C. A B C D
 D. A B C D

✅ **Answer: A B C D** (Base `A` constructed once)

---

22. What problem is solved by virtual base class?

A. Memory leak
 B. Ambiguity
 C. Overloading
 D. Overriding

✅ **Answer:** B

---

23. Output?

```
class A {
public:
 A() { cout << "A"; }
};
class B : public A {
public:
 B() { cout << "B"; }
};
class C : public B {
public:
 C() { cout << "C"; }
};
int main() {
 C obj;
}
```

A. ABC
 B. CBA
 C. ACB
 D. Error

✅ **Answer:** A

---

24. Which inheritance is NOT directly supported in C++?

A. Single
 B. Multiple
 C. Multilevel
 D. Cyclic

✅ **Answer:** D

---

25. Output?

```
class A {
public:
 int x = 10;
};
class B : public A {};
class C : public A {};
class D : public B, public C {};
int main() {
 D obj;
 cout << obj.x;
}
```

A. 10
 B. Error (ambiguous)
 C. 0
 D. Garbage

✅ **Answer:** B
(diamond problem without virtual)

---

26. Which inheritance causes diamond problem?

A. Single
B. Multiple
C. Multilevel
D. Hierarchical

✅ **Answer:** B

---

27. Friend function in inheritance

A. Is inherited
B. Is not inherited
C. Becomes member
D. Is virtual

✅ **Answer:** B

---

28. Output?

```
class Printer {
public:
 void print() { cout << "Printer"; }
};
class Scanner {
public:
 void scan() { cout << "Scanner"; }
};
class AllInOne : public Printer, public Scanner {};
int main() {
 AllInOne obj;
 obj.print();
 obj.scan();
}
```

A. PrinterScanner
B. ScannerPrinter
C. Error
D. Printer

✅ **Answer:** A
(multiple inheritance)

---

29. Which inheritance is best for printer hierarchy?

A. Single
B. Multiple
C. Hybrid
D. Hierarchical

✅ **Answer:** B *(printer + scanner + fax)*

---

30. Destructor order in inheritance

A. Base → Derived
B. Derived → Base
C. Random
D. Compiler dependent

✅ **Answer:** B

Session 11

🟢 EASY MCQs (1–20)

---

1. Polymorphism means

A. One class
B. One function
C. Many forms
D. Many classes

✅ **Answer:** C

---

2. Which is compile-time polymorphism?

A. Virtual function
B. Function overloading
C. Inheritance
D. Dynamic binding

✅ **Answer:** B

---

3. Operator overloading is

A. Creating new operator
B. Giving new meaning to existing operator
C. Removing operator
D. Hiding operator

✅ **Answer:** B

---

4. Which operator cannot be overloaded?

A. +
B. ==
C. ::
D. <<

✅ **Answer:** C

---

5. Function overloading depends on

A. Return type
B. Function name
C. Number/type of parameters
D. Scope

✅ **Answer:** C

---

6. Output?

```
int add(int a, int b) { return a+b; }
int add(int a, int b, int c) { return a+b+c; }
cout << add(2,3);
```

A. 5
B. 6
C. Error
D. Garbage

✅ **Answer:** A

---

7. Which operator is overloaded for output?

A. >>
B. <<
C. =
D. []

✅ **Answer:** B

---

8. Friend function

A. Is member of class
B. Can access private members
C. Is inherited
D. Must be virtual

✅ **Answer:** B

---

9. Which keyword is used to make function constant?

A. static
B. const
C. final
D. inline

✅ **Answer:** B

---

10. Constant member function means

A. Function is constant
B. Object cannot change data
C. Function cannot be called
D. Object is constant

✅ **Answer:** B

---

11. Output?

```
class A {
public:
 int x=10;
 void show() const {
  cout << x;
 }
};
```

A. Error
B. 10
C. Garbage
D. 0

✅ **Answer:** B

---

12. Which operator is overloaded for input?

A. <<
B. >>
C. []
D. ++

✅ **Answer:** B

---

13. Operator overloading is done using

A. operator keyword
B. overload keyword

C. friend keyword
D. virtual keyword

✅ **Answer:** A

---

14. Which operator is unary?

A. +
B. ++
C. ==
D. []

✅ **Answer:** B

---

15. Output?

```
class Test {
public:
 int x;
 Test(int a):x(a){}
};
```

A. Default constructor
B. Parameterized constructor
C. Copy constructor
D. Error

✅ **Answer:** B

---

16. Which polymorphism is resolved at runtime?

A. Function overloading
B. Operator overloading
C. Compile-time
D. Runtime

✅ **Answer:** D

---

17. Which operator is used to access array elements?

A. ()
B. []
C. {}
D. <>

✅ **Answer:** B

---

18. Output?

```
int x=5;
cout << ++x;
```

A. 5
B. 6
C. Garbage
D. Error

✅ **Answer:** B

---

19. Which operator compares equality?

A. =
B. !=
C. ==
D. <=

✅ **Answer:** C

---

20. Which feature allows same function name?

A. Encapsulation
B. Inheritance
C. Polymorphism
D. Abstraction

✅ **Answer:** C

🟡 MEDIUM MCQs (21–40)

---

21. Output?

```
class A {
public:
 void fun(int x) { cout<<"Int"; }
 void fun(double x) { cout<<"Double"; }
};
int main() {
 A obj;
 obj.fun(10.5);
}
```

A. Int
B. Double
C. Error
D. Garbage

✅ **Answer:** B

---

22. Which operator overloading must be member function?

A. +
B. ==
C. []
D. <<

✅ **Answer:** C

---

23. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 friend void show(A a);
};
void show(A a) {
 cout << a.x;
}
int main() {
 A obj(10);
 show(obj);
}
```

A. Error
B. 10
C. Garbage
D. 0

✅ **Answer:** B

---

24. Which operator cannot be friend function?

A. <<
B. >>
C. =
D. +

✅ **Answer:** C

---

25. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 A operator+(A b) {
  return A(x + b.x);
 }
};
int main() {
 A a(5), b(10);
 A c = a + b;
 cout << 15;
}
```

A. 5
B. 10
C. 15
D. Error

✅ **Answer:** C

---

26. Which operator returns object reference?

A. +
B. =
C. ==
D. <<

✅ **Answer:** B

---

27. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 bool operator==(A b) {
  return x == b.x;
 }
};
int main() {
 A a(5), b(5);
 cout << (a == b);
}
```

A. true
B. false
C. 1
D. 0

✅ **Answer:** C

---

28. Which operator overloading supports chaining?

A. ==
B. =
C. []
D. ++

✅ **Answer:** B

---

29. Constant member functions cannot

A. Access private members
B. Modify data members
C. Be overloaded
D. Be friend

✅ **Answer:** B

---

30. Output?

```
class A {
public:
 int x=10;
};
int main() {
 const A a;
 cout << a.x;
}
```

A. Error
B. 10
C. Garbage
D. 0

✅ **Answer:** B

---

31. Which operator is binary?

A. ++
B. --
C. +
D. !

✅ **Answer:** C

---

32. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 A& operator++() {
  ++x;
  return *this;
 }
 int get(){return x;}
};
int main() {
 A a(5);
 ++a;
 cout << a.get();
}
```

A. 5
B. 6
C. Error
D. Garbage

✅ **Answer:** B

---

33. Which operator is overloaded for indexing?

A. ()
B. []
C. {}
D. <>

✅ **Answer:** B

---

34. Operator << is usually overloaded as

A. Member
B. Friend
C. Static
D. Inline

✅ **Answer:** B

---

35. Output?

```
class A {
public:
 int operator[](int i) { return i*2; }
};
int main() {
 A a;
 cout << a[3];
}
```

A. 3
B. 6
C. Error
D. Garbage

✅ **Answer:** B

---

36. Function overloading ignores

A. Parameter count
B. Parameter type
C. Return type only
D. Function name

✅ **Answer:** C

---

37. Which operator cannot be overloaded?

A. sizeof
B. +
C. []
D. <<

✅ **Answer:** A

---

38. Output?

```
class A {
public:
 A operator++(int) {
  return *this;
 }
};
```

A. Prefix ++
B. Postfix ++
C. Error
D. Unary +

✅ **Answer:** B

---

39. Friend function is declared using

A. public
B. private
C. friend
D. virtual

✅ **Answer:** C

---

40. Which supports polymorphism most directly here?

A. Encapsulation
B. Inheritance
C. Function overloading
D. Namespace

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (41–60)

41. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 A& operator=(const A& a) {
  x = a.x;
  return *this;
 }
};
```

This supports:
 A. Copying
 B. Chaining
 C. Comparison
 D. Overloading

✅ **Answer:** B

---

42. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 friend ostream& operator<<(ostream& os, A a) {
  os << a.x;
  return os;
 }
};
int main() {
 A a(10);
 cout << a;
}
```

A. Error
 B. 10

C. Garbage
D. 0

✅ **Answer:** B

---

43. Which operator overloading is dangerous if misused?

A. +
B. []
C. =
D. <<

✅ **Answer:** C

---

44. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 bool operator==(A& a) const {
  return x == a.x;
 }
};
```

Which function is const?
A. operator==
B. x
C. a
D. return

✅ **Answer:** A

---

45. Which operator overloading enables array-like access?

A. ()
B. []
C. {}
D. <>

✅ **Answer:** B

---

46. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 A operator--() {
  return A(--x);
 }
};
```

This is:
A. Postfix --
B. Prefix --
C. Binary --
D. Error

✅ **Answer:** B

---

47. Which operator must be member function?

A. <<
B. >>
C. []
D. +

✅ **Answer:** C

---

48. Friend functions break

A. Polymorphism
B. Encapsulation
C. Inheritance
D. Abstraction

✅ **Answer:** B

---

49. Which operator cannot be overloaded?

A. ?:
B. +
C. []
D. ==

✅ **Answer:** A

---

50. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 int operator[](int i) const {
  return x + i;
 }
};
int main() {
 const A a(5);
 cout << a[2];
}
```

A. Error
B. 5
C. 7
D. Garbage

✅ **Answer:** C

---

51. Overloading << and >> usually requires

A. Member functions
B. Friend functions
C. Inline functions
D. Static functions

✅ **Answer:** B

---

52. Which operator overloading supports IO chaining?

A. +
B. ==
C. <<
D. []

✅ **Answer:** C

---

53. Constant function ensures

A. Function const
B. Object const
C. Data not modified
D. Operator const

✅ **Answer:** C

---

54. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 A operator+(int y) {
  return A(x+y);
 }
};
int main() {
 A a(5);
 A b = a + 10;
 cout << 15;
}
```

A. 5
B. 10
C. 15
D. Error

✅ **Answer:** C

55. Best use of operator overloading

A. Make code confusing
B. Mimic built-in behavior
C. Replace functions
D. Avoid classes

✅ **Answer:** B

---

56. Which operator returns reference for chaining?

A. +
B. ==
C. =
D. []

✅ **Answer:** C

---

57. Operator [] should return

A. Object
B. Pointer
C. Reference
D. Value only

✅ **Answer:** C *(best practice)*

---

58. Output?

```
class A {
 int x;
public:
 A(int a):x(a){}
 friend bool operator==(A a, A b) {
  return a.x == b.x;
 }
};
int main() {
 A a(5), b(5);
 cout << (a == b);
}
```

A. true
B. false
C. 1
D. 0

✅ **Answer:** C

---

59. Operator overloading happens at

A. Runtime
B. Compile time
C. Link time
D. Execution

✅ **Answer:** B

---

60. Polymorphism achieved here is

A. Runtime
B. Dynamic
C. Compile-time
D. Virtual

✅ **Answer:** C

Session 13

🟢 EASY MCQs (1–10)

---

1. What is an exception in C++?

A. Syntax error
B. Logical error

C. Runtime abnormal condition
D. Compile-time warning

✅ **Answer:** C

---

2. Which keyword is used to throw an exception?

A. catch
B. throw
C. try
D. exception

✅ **Answer:** B

---

3. Which block handles exceptions?

A. try
B. throw
C. catch
D. finally

✅ **Answer:** C

---

4. Output?

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

A. 0
B. 10
C. Error
D. Nothing

✅ **Answer:** B

---

5. try block must be followed by

A. finally
B. throw
C. catch
D. return

✅ **Answer:** C

---

6. Which header defines standard exception class?

A. <iostream>
B. <exception>
C. <stdexcept>
D. <error>

✅ **Answer:** B

---

7. Output?

```
try {
 cout << "A";
}
catch(...) {
 cout << "B";
}
```

A. A
B. B
C. AB
D. Error

✅ **Answer:** A *(no exception thrown)*

---

8. Which catch block catches all exceptions?

A. catch(int)
B. catch(char*)
C. catch(...)
D. catch(exception)

✅ **Answer:** C

---

9. Exception handling is used to

A. Avoid errors
B. Handle runtime errors
C. Speed up code
D. Replace if-else

✅ **Answer:** B

---

10. Custom exception classes usually inherit from

A. error
B. exception
C. runtime
D. object

✅ **Answer:** B

🟡 MEDIUM MCQs (11–20)

---

11. Output?

```
try {
 throw 5.5;
}
catch(int x) {
 cout << "Int";
}
catch(double x) {
 cout << "Double";
}
```

A. Int
B. Double
C. Error
D. Nothing

✅ **Answer:** B

---

12. Order of catch blocks should be

A. Any order
B. Derived first, base later
C. Base first, derived later
D. Alphabetical

✅ **Answer:** B

---

13. Output?

```
try {
 throw 10;
}
catch(char c) {
 cout << "Char";
}
catch(...) {
 cout << "All";
}
```

A. Char
B. All
C. 10
D. Error

✅ **Answer:** B

---

14. Which statement rethrows an exception?

A. throw;
B. throw();
C. return;
D. continue;

✅ **Answer:** A

---

15. Output?

```
try {
 try {
  throw 1;
 }
 catch(int x) {
  cout << "Inner ";
  throw;
 }
}
catch(int x) {
 cout << "Outer";
}
```

A. Inner
B. Outer
C. Inner Outer
D. Error

✅ **Answer:** C

---

16. Which is TRUE?

A. catch must always be present
B. Multiple catch blocks allowed
C. try can exist alone
D. throw must be inside catch

✅ **Answer:** B

---

17. Output?

```
try {
 throw "Error";
}
catch(const char* e) {
 cout << e;
}
```

A. Error
B. Garbage
C. Compilation error
D. Nothing

✅ **Answer:** A

---

18. Which type of exception is preferred in C++?

A. int
B. char*
C. class object
D. float

✅ **Answer:** C

---

19. Exception handling affects

A. Compile time only
B. Runtime only
C. Both
D. Neither

✅ **Answer:** B

---

20. What happens if exception is not caught?

A. Ignored
B. Program continues
C. Program terminates
D. Compiler handles

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
try {
 throw 10;
}
catch(double x) {
 cout << "Double";
}
```

A. Double
B. 10
C. Runtime terminate
D. Compilation error

✅ **Answer:** C *(no matching catch)*

---

22. Output?

```
try {
 throw 5;
}
catch(int x) {
 cout << "A";
 throw;
}
catch(...) {
 cout << "B";
}
```

A. A
B. B
C. AB
D. Program terminates

✅ **Answer:** D *(rethrown, no outer handler)*

---

23. Which function is called when exception is uncaught?

A. exit()
B. abort()
C. terminate()
D. stop()

✅ **Answer:** C

---

24. Output?

```
try {
 cout << "X";
 throw 1;
 cout << "Y";
}
catch(int) {
 cout << "Z";
}
```

A. XYZ
B. XZ
C. Z
D. XY

✅ **Answer:** B

---

25. Which is TRUE about exception objects?

A. Must be primitive
B. Cannot be class
C. Can be user-defined class
D. Must be pointer

✅ **Answer:** C

---

26. Output?

```
class MyEx {};
try {
 throw MyEx();
}
catch(MyEx e) {
 cout << "Handled";
}
```

A. Error
B. Handled
C. Garbage
D. Nothing

✅ **Answer:** B

---

27. Best way to catch exceptions is

A. By value
B. By pointer
C. By reference
D. By macro

✅ **Answer:** C

---

28. Output?

```
try {
 throw 10;
}
catch(const int& x) {
 cout << x;
}
```

A. Error
B. 0
C. 10
D. Garbage

✅ **Answer:** C

---

29. Which is deprecated in modern C++?

A. try-catch
B. throw keyword
C. exception specification (throw())
D. custom exception

✅ **Answer:** C

---

30. Custom exception class should ideally

A. Be empty
B. Inherit from std::exception
C. Use printf
D. Avoid constructors

✅ **Answer:** B

Session 12

🟢 EASY MCQs (1–10)

---

1. Runtime polymorphism is achieved using

A. Function overloading
B. Operator overloading
C. Virtual functions
D. Templates

✅ **Answer:** C

---

2. A virtual function is

A. Inline function
B. Static function
C. Function resolved at runtime
D. Compile-time function

✅ **Answer:** C

---

3. Which keyword is used to declare virtual function?

A. abstract
B. dynamic

C. virtual
D. override

✅ **Answer:** C

---

4. Output?

```
class A {
public:
 virtual void show() { cout << "A"; }
};
class B : public A {
public:
 void show() { cout << "B"; }
};
int main() {
 A* p = new B();
 p->show();
}
```

A. A
B. B
C. Error
D. Garbage

✅ **Answer:** B

---

5. A pure virtual function is declared using

A. `=1`
B. `=0`
C. `==0`
D. `virtual` only

✅ **Answer:** B

---

6. A class containing at least one pure virtual function is

A. Concrete class
B. Derived class
C. Abstract class
D. Interface

✅ **Answer:** C

---

7. Can we create object of abstract class?

A. Yes
B. No
C. Only using pointer
D. Only using reference

✅ **Answer:** B

---

8. Which cast is safest for downcasting?

A. static_cast
B. reinterpret_cast
C. dynamic_cast
D. const_cast

✅ **Answer:** C

---

9. Interface in C++ is implemented using

A. Class with static members
B. Class with only data members
C. Class with all pure virtual functions
D. Namespace

✅ **Answer:** C

---

10. Virtual destructor is needed when

A. Class has no data
B. Base pointer deletes derived object
C. Only stack objects used
D. No inheritance

✅ **Answer:** B

🟡 MEDIUM MCQs (11–20)

---

11. Output?

```
class Base {
public:
 void show() { cout << "Base"; }
};
class Derived : public Base {
public:
 void show() { cout << "Derived"; }
};
int main() {
 Base* b = new Derived();
 b->show();
}
```

A. Base
B. Derived
C. Error
D. Garbage

✅ **Answer:** A *(no virtual function)*

---

12. Which is TRUE about pure virtual function?

A. Must have body
B. Cannot have body
C. May have body
D. Cannot be overridden

✅ **Answer:** C

---

13. Output?

```
class A {
```

```
public:
 virtual ~A() { cout << "A"; }
};
class B : public A {
public:
 ~B() { cout << "B"; }
};
int main() {
 A* p = new B();
 delete p;
}
```

A. A
B. B
C. BA
D. AB

✅ **Answer:** C *(derived → base)*

---

14. Which cast removes constness?

A. static_cast
B. dynamic_cast
C. const_cast
D. reinterpret_cast

✅ **Answer:** C

---

15. Output?

```
class Printer {
public:
 virtual void print() = 0;
};
class Inkjet : public Printer {
public:
 void print() { cout << "Inkjet"; }
};
int main() {
 Printer* p = new Inkjet();
 p->print();
}
```

A. Printer
B. Inkjet
C. Error
D. Garbage

✅ **Answer:** B

---

16. Which cast performs compile-time conversion?

A. dynamic_cast
B. static_cast
C. reinterpret_cast
D. const_cast

✅ **Answer:** B

---

17. Diamond problem occurs in

A. Single inheritance
B. Multilevel inheritance
C. Multiple inheritance
D. Hierarchical inheritance

✅ **Answer:** C

---

18. Output?

```
class A {
public:
 virtual void f() { cout<<"A"; }
};
class B : public A {
public:
 void f() { cout<<"B"; }
};
```

```
class C : public B {};
int main() {
 A* p = new C();
 p->f();
}
```

A. A
B. B
C. C
D. Error

✅ **Answer:** B

---

19. Which keyword avoids duplicate base class in diamond problem?

A. static
B. friend
C. virtual
D. protected

✅ **Answer:** C

---

20. Abstract class is mainly used for

A. Object creation
B. Data storage
C. Defining interface
D. Performance

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
class A {
public:
 virtual void show() { cout<<"A"; }
};
class B : virtual public A {};
class C : virtual public A {};
class D : public B, public C {};
int main() {
 D obj;
 obj.show();
}
```

A. A
B. Error
C. Ambiguous
D. Garbage

✅ **Answer:** A *(virtual base class)*

---

22. Output?

```
class A {
public:
 virtual void f() = 0;
};
class B : public A {};
int main() {
 B obj;
}
```

A. Compiles
B. Runtime error
C. Compilation error
D. Garbage

✅ **Answer:** C *(B still abstract)*

---

23. Which cast can fail at runtime?

A. static_cast
B. const_cast

C. dynamic_cast
D. reinterpret_cast

✅ **Answer:** C

---

24. Output?

```
class A {
public:
 virtual void show() { cout<<"A"; }
};
class B : public A {
public:
 void show() { cout<<"B"; }
};
int main() {
 B b;
 A& ref = b;
 ref.show();
}
```

A. A
B. B
C. Error
D. Garbage

✅ **Answer:** B

---

25. Which function call is resolved at runtime?

A. Inline
B. Overloaded
C. Virtual
D. Static

✅ **Answer:** C

---

26. Output?

```
class A {
```

```
public:
 virtual void f() { cout<<"A"; }
};
class B : public A {
public:
 void f() { cout<<"B"; }
};
int main() {
 A a;
 B b;
 A* p = &b;
 p->f();
}
```

A. A
B. B
C. Error
D. Garbage

✅ **Answer:** B

---

27. reinterpret_cast is mainly used for

A. Safe type conversion
B. Downcasting
C. Low-level pointer conversion
D. Removing const

✅ **Answer:** C

---

28. Output?

```
class Printer {
public:
 virtual void print() = 0;
};
class Scanner {
public:
 virtual void scan() = 0;
};
class AllInOne : public Printer, public Scanner {
public:
 void print() { cout<<"Print "; }
 void scan() { cout<<"Scan"; }
};
int main() {
 AllInOne a;
 a.print();
 a.scan();
}
```

A. PrintScan
B. ScanPrint
C. Error
D. Garbage

✅ **Answer:** A

---

29. Which is TRUE about interface in C++?

A. Uses keyword interface
B. Contains only static methods
C. Uses abstract class with pure virtual functions
D. Cannot be inherited

✅ **Answer:** C

---

30. Best real-life example of diamond problem

A. Vehicle → Car
B. Printer → Inkjet
C. Person → Student → Employee
D. Person → Student, Person → Employee → TeachingAssistant

✅ **Answer:** D

Session 14

🟢 EASY MCQs (1–10)

---

1. Which header file is required for C++ I/O streams?

A. `<stdio.h>`
B. `<fstream>`
C. `<iostream>`
D. `<stream>`

✅ **Answer:** C

---

2. `cin` is an object of which class?

A. ostream
B. istream
C. iostream
D. stream

✅ **Answer:** B

---

3. `cout` belongs to

A. istream
B. ostream
C. fstream
D. iostream

✅ **Answer:** B

---

4. Output?

```
int x = 10;
cout << x;
```

A. x
B. 10
C. Garbage
D. Error

✅ **Answer:** B

---

5. Which operator is used with `cin`?

A. `<<`
B. `>>`
C. `->`
D. `.`

✅ **Answer:** B

---

6. Which operator is used with `cout`?

A. `>>`
B. `<<`
C. `==`
D. `=`

✅ **Answer:** B

---

7. `endl` does

A. Prints newline only
B. Flushes buffer only
C. Prints newline and flushes buffer
D. Clears screen

✅ **Answer:** C

---

8. Which is an unformatted I/O function?

A. `setw()`
B. `setprecision()`

C. `get()`
D. `fixed`

✅ **Answer:** C

---

9. Which namespace contains stream objects?

A. global
B. std
C. io
D. stream

✅ **Answer:** B

---

10. Output?

```
cout << "Hello" << endl << "World";
```

A. HelloWorld
B. Hello World
C. Hello
World
D. Error

✅ **Answer:** C

🟡 MEDIUM MCQs (11–20)

---

11. Which class is base of all stream classes?

A. istream
B. ostream
C. ios
D. iostream

✅ **Answer:** C

---

12. Output?

```
cout << setw(5) << 10;
```

A. 10
B. 10
C. 00010
D. Error

✅ **Answer:** B

---

13. Which header is required for manipulators?

A. `<iostream>`
B. `<iomanip>`
C. `<stream>`
D. `<ios>`

✅ **Answer:** B

---

14. Output?

```
cout << setprecision(3) << 3.14159;
```

A. 3.14
B. 3.142
C. 3.1
D. 3

✅ **Answer:** C *(default precision = total digits)*

---

15. Which manipulator forces fixed decimal notation?

A. setw
B. setprecision
C. fixed
D. scientific

✅ **Answer:** C

---

16. Output?

```
cout << fixed << setprecision(2) << 3.14159;
```

A. 3.14
B. 3.1416
C. 3.1
D. 3

✅ **Answer:** A

---

17. Which function reads a single character including whitespace?

A. cin >> ch
B. getline()
C. get()
D. read()

✅ **Answer:** C

---

18. Output?

```
char ch;
cin.get(ch);
cout << ch;
```

Input: `A`
A. A
B. ASCII of A
C. Garbage
D. Error

✅ **Answer:** A

---

19. Which function reads an entire line?

A. cin >>
B. cin.get()
C. getline(cin, str)
D. read()

✅ **Answer:** C

---

20. Which manipulator resets width after use?

A. setw
B. fixed
C. endl
D. setprecision

✅ **Answer:** A

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
cout << setw(5) << setfill('*') << 10;
```

A. 10
B. ***10
C. **010
D. Error

✅ **Answer:** B

---

22. Output?

```
cout << setprecision(2) << fixed << 12.3456;
```

A. 12
B. 12.3
C. 12.35
D. 12.3456

✅ **Answer:** C

---

23. Which I/O is faster?

A. cin/cout
B. scanf/printf
C. Depends on sync
D. Always cin/cout

✅ **Answer:** C

---

24. Output?

```
int x = 10;
cout << showbase << hex << x;
```

A. a
B. 0xa
C. 10
D. Error

✅ **Answer:** B

---

25. Which manipulator displays base prefix (0x, 0)?

A. showpos
B. showbase
C. setw
D. fixed

✅ **Answer:** B

---

26. Output?

```
cout << boolalpha << (10 > 5);
```

A. 1
B. true
C. false
D. Error

✅ **Answer:** B

---

27. Unformatted input function

```
cin.read(buffer, 5);
```

This reads:
A. Until newline
B. Exactly 5 characters
C. One word
D. One line

✅ **Answer:** B

---

28. Output?

```
cout << noshowpos << showpos << 10;
```

A. 10
B. +10
C. ++10
D. Error

✅ **Answer:** B

---

29. Which stream is used for error output?

A. cin
B. cout
C. cerr
D. clog

✅ **Answer:** C

---

30. Output?

```
cerr << "Error";
```

A. Printed to file
B. Printed to console immediately
C. Buffered output
D. No output

✅ **Answer:** B *(unbuffered stream)*

Session 15

🟢 EASY MCQs (1–10)

---

1. What is a file?

A. Collection of functions
B. Collection of classes
C. Collection of data stored permanently
D. Temporary memory

✅ **Answer:** C

---

2. Which header file is required for file handling in C++?

A. `<iostream>`
B. `<fstream>`
C. `<file>`
D. `<stream>`

✅ **Answer:** B

---

3. Which class is used to write data to a file?

A. ifstream
B. ofstream
C. fstream
D. ostream

✅ **Answer:** B

---

4. Which class is used to read data from a file?

A. ofstream
B. ifstream
C. fstream
D. istream

✅ **Answer:** B

---

5. Which operator is used to write data into file?

A. `>>`
B. `<<`
C. `->`
D. `==`

✅ **Answer:** B

---

6. Output?

```
ofstream fout("data.txt");
fout << "Hello";
fout.close();
```

A. Prints Hello
B. Stores Hello in file
C. Error
D. Nothing happens

✅ **Answer:** B

---

7. Which mode opens file for reading only?

A. ios::out
B. ios::in
C. ios::app
D. ios::binary

✅ **Answer:** B

---

8. Which mode appends data at end of file?

A. ios::out
B. ios::in
C. ios::app
D. ios::ate

✅ **Answer:** C

---

9. File pointer initially points to

A. End of file
B. Middle of file
C. Beginning of file
D. Random location

✅ **Answer:** C

---

10. Which function closes a file?

A. end()
B. stop()
C. close()
D. exit()

✅ **Answer:** C

🟡 MEDIUM MCQs (11–20)

---

11. Output?

```
ifstream fin("data.txt");
string s;
fin >> s;
cout << s;
```

(Assume file contains: `Hello World`)
A. Hello
B. World
C. Hello World
D. Error

✅ **Answer:** A *(>> reads till whitespace)*

---

12. Which class can both read and write file?

A. ifstream
B. ofstream
C. fstream
D. iostream

✅ **Answer:** C

---

13. Which mode truncates existing file?

A. ios::app
B. ios::ate
C. ios::out
D. ios::binary

✅ **Answer:** C

---

14. Output?

```
ofstream fout("test.txt");
fout << 10 << " " << 20;
```

A. 1020
B. 10 20
C. Error
D. Garbage

✅ **Answer:** B

---

15. Which function checks end-of-file?

A. eof()
B. end()
C. finish()
D. last()

✅ **Answer:** A

---

16. Output?

```
ifstream fin("data.txt");
char ch;
while(fin.get(ch))
 cout << ch;
```

This prints:
A. First word only
B. Whole file character by character
C. Nothing
D. Error

✅ **Answer:** B

---

17. Which mode opens file in binary form?

A. ios::binary
B. ios::in
C. ios::out
D. ios::app

✅ **Answer:** A

---

18. Which function moves file pointer?

A. seekg()
B. seekp()
C. Both A and B
D. move()

✅ **Answer:** C

---

19. Output?

```
ofstream fout("data.txt", ios::app);
fout << "C++";
```

A. Overwrites file
B. Deletes file
C. Appends C++
D. Error

✅ **Answer:** C

---

20. Which is TRUE?

A. Files are temporary
B. File data is lost after program ends
C. Files provide permanent storage
D. Files exist only in RAM

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
ofstream fout("data.txt");
fout << "ABC";
fout.close();

ifstream fin("data.txt");
char ch;
fin >> ch;
cout << ch;
```

A. ABC
B. A
C. Error
D. Garbage

✅ **Answer:** B

---

22. Which mode opens file and moves pointer to end but allows overwrite?

A. ios::app
B. ios::ate
C. ios::out
D. ios::binary

✅ **Answer:** B

---

23. Output?

```
ifstream fin("data.txt");
string s;
getline(fin, s);
cout << s;
```

(File contains: `Hello World`)
A. Hello
B. World
C. Hello World
D. Error

✅ **Answer:** C

---

24. Which is NOT a valid file mode?

A. ios::in
B. ios::out
C. ios::read
D. ios::app

✅ **Answer:** C

---

25. Output?

```
fstream file("data.txt", ios::out | ios::in);
file << "Hi";
file.seekg(0);
string s;
file >> s;
cout << s;
```

A. Hi
B. Garbage
C. Error
D. Empty

✅ **Answer:** A

---

26. If file does not exist and ios::in is used

A. File created
B. File opened
C. Error occurs
D. File overwritten

✅ **Answer:** C

---

27. Which function returns current position of get pointer?

A. tellp()
B. tellg()
C. seekg()
D. seekp()

✅ **Answer:** B

---

28. Output?

```
ofstream fout("data.txt");
fout << "C++ File";
fout.close();

ifstream fin("data.txt");
char ch;
fin.get(ch);
cout << ch;
```

A. C
B. +
C. File
D. Error

✅ **Answer:** A

---

29. Which is best for reading binary files?

A. ifstream
B. ofstream
C. fstream with ios::binary
D. cout

✅ **Answer:** C

---

30. Best practice after file operation

A. Ignore file
B. Close file
C. Delete file
D. Flush console

✅ **Answer:** B

Session 16

🟢 EASY MCQs (1–10)

---

1. What is a template in C++?

A. A macro
B. A generic blueprint for functions/classes
C. A class only
D. A library

✅ **Answer:** B

---

2. Templates support

A. Runtime polymorphism
B. Compile-time polymorphism
C. Dynamic binding
D. Virtual functions

✅ **Answer:** B

---

3. Which keyword is used to define template?

A. generic
B. class
C. template
D. typename

✅ **Answer:** C

---

4. Syntax of function template

```
template <typename T>
T fun(T a, T b);
```

`T` represents:
A. Class
B. Variable
C. Data type
D. Object

✅ **Answer:** C

---

5. Output?

```
template <class T>
T add(T a, T b) {
 return a + b;
}
int main() {
 cout << add(2,3);
}
```

A. 5
B. 23
C. Error
D. Garbage

✅ **Answer:** A

---

6. Template functions are expanded at

A. Runtime
B. Compile time
C. Link time
D. Execution time

✅ **Answer:** B

---

7. Which is correct?

A. template <T>
B. template <class T>
C. template (T)
D. template [T]

✅ **Answer:** B

---

8. Which feature avoids code duplication?

A. Inheritance
B. Templates
C. Polymorphism
D. Macros

✅ **Answer:** B

---

9. Templates work with

A. Only primitive types
B. Only classes
C. Any data type
D. Only int

✅ **Answer:** C

---

10. Templates are type-safe compared to

A. Classes
B. Functions
C. Macros
D. Namespaces

✅ **Answer:** C

🟡 MEDIUM MCQs (11–20)

---

11. Output?

```
template <typename T>
T max(T a, T b) {
 return (a > b) ? a : b;
}
int main() {
 cout << max(10, 20);
}
```

A. 10
B. 20
C. Error
D. Garbage

✅ **Answer:** B

---

12. Output?

```
template <class T>
void fun(T x) {
 cout << x;
}
int main() {
 fun(10.5);
}
```

A. 10
B. 10.5
C. Error
D. Garbage

✅ **Answer:** B

---

13. Which template supports multiple data types?

A. Function template
B. Class template
C. Both
D. None

✅ **Answer:** C

---

14. Output?

```
template <class T>
class Test {
 T x;
public:
 Test(T a):x(a){}
 void show() { cout << x; }
};
int main() {
 Test<int> t(10);
 t.show();
}
```

A. 0
B. 10
C. Error
D. Garbage

✅ **Answer:** B

---

15. Which is TRUE about templates?

A. Cannot be overloaded
B. Cannot be specialized
C. Are expanded by compiler
D. Work only for functions

✅ **Answer:** C

---

16. Output?

```
template <class T>
T square(T x) {
 return x * x;
}
int main() {
 cout << square(2.5);
}
```

A. 5
B. 6.25
C. Error
D. Garbage

✅ **Answer:** B

---

17. Which symbol indicates template specialization?

A. <>
B. ()
C. {}
D. []

✅ **Answer:** A

---

18. Which template parameter is valid?

A. int
B. class

C. typename
D. Both B and C

✅ **Answer:** D

---

19. Templates generate

A. Single function
B. Multiple functions for types
C. Runtime code
D. Dynamic memory

✅ **Answer:** B

---

20. Class template object creation

A. `Test t;`
B. `Test<int> t;`
C. `template t;`
D. `Test t<int>;`

✅ **Answer:** B

🔴 HARD & TRICKY MCQs (21–30)

---

21. Output?

```
template <class T>
void fun(T x) {
 cout << "Generic";
}
void fun(int x) {
 cout << "Int";
}
int main() {
 fun(10);
}
```

A. Generic
B. Int
C. Error
D. Garbage

✅ **Answer:** B
(Exact match preferred over template)

---

22. Output?

```
template <class T>
T add(T a, T b) {
 return a + b;
}
int main() {
 cout << add(2, 3.5);
}
```

A. 5
B. 5.5
C. Error
D. Garbage

✅ **Answer:** C
(type mismatch)

---

23. Which template argument is NOT allowed?

A. typename
B. class
C. int
D. pointer

✅ **Answer:** C *(int without non-type syntax)*

---

24. Output?

```
template <class T>
class A {
public:
 static int x;
};
template <class T>
int A<T>::x = 5;
int main() {
 A<int> a;
 cout << a.x;
}
```

A. 0
B. 5
C. Error
D. Garbage

✅ **Answer:** B

---

25. Which is TRUE?

A. Templates increase runtime overhead
B. Templates are resolved at runtime
C. Templates may increase code size
D. Templates reduce compilation

✅ **Answer:** C

---

26. Output?

```
template <class T>
T fun(T a, T b) {
 return a + b;
}
int main() {
 cout << fun('A','B');
}
```

A. AB
B. 131
C. Error
D. Garbage

✅ **Answer:** B
(ASCII addition)

---

27. Template specialization means

A. Using same template
B. Creating specific implementation
C. Overloading function
D. Inheritance

✅ **Answer:** B

---

28. Which is NOT a drawback of templates?

A. Code bloat
B. Longer compile time
C. Type safety
D. Complex error messages

✅ **Answer:** C

---

29. Output?

```
template <class T>
class Test {
public:
 void show() { cout << "Generic"; }
};
template <>
class Test<int> {
public:
 void show() { cout << "Int"; }
};
int main() {
 Test<int> t;
 t.show();
}
```

A. Generic
B. Int
C. Error
D. Garbage

✅ **Answer:** B

---

30. Templates provide which type of polymorphism?

A. Runtime
B. Dynamic
C. Compile-time
D. Virtual

✅ **Answer:** C

Session 17 & 18

🟢 EASY MCQs (1–20)

---

1. STL stands for

A. Standard Type Library
B. System Template Library
C. Standard Template Library
D. Structured Template Library

✅ **Answer:** C

---

2. STL mainly consists of

A. Containers only
B. Algorithms only
C. Containers, Algorithms, Iterators
D. Functions only

✅ **Answer:** C

---

3. Which header is required for vector?

A. `<array>`
B. `<vector>`
C. `<list>`
D. `<container>`

✅ **Answer:** B

---

4. Which container allows dynamic resizing?

A. array
B. vector
C. stack
D. map

✅ **Answer:** B

---

5. Output?

```
vector<int> v = {1,2,3};
cout << v.size();
```

A. 2
B. 3
C. 4
D. Error

✅ **Answer:** B

---

6. Stack follows

A. FIFO
B. LIFO
C. Random
D. Priority

✅ **Answer:** B

---

7. Which header is needed for stack?

A. `<stack>`
B. `<queue>`
C. `<vector>`
D. `<map>`

✅ **Answer:** A

---

8. Queue follows

A. LIFO
B. FIFO
C. Random
D. Sorted

✅ **Answer:** B

---

9. Which container stores key-value pairs?

A. vector
B. stack
C. map
D. queue

✅ **Answer:** C

---

10. Output?

```
stack<int> s;
s.push(10);
s.push(20);
cout << s.top();
```

A. 10
B. 20

C. Error
D. Garbage

✅ **Answer:** B

---

11. Which function inserts element at end of vector?

A. insert()
B. add()
C. push_back()
D. push()

✅ **Answer:** C

---

12. Output?

```
queue<int> q;
q.push(1);
q.push(2);
cout << q.front();
```

A. 1
B. 2
C. Error
D. Garbage

✅ **Answer:** A

---

13. Map stores keys

A. In insertion order
B. In sorted order
C. Random order
D. Reverse order

✅ **Answer:** B

---

14. Which header is needed for map?

A. `<map>`
B. `<unordered_map>`
C. `<pair>`
D. `<set>`

✅ **Answer:** A

---

15. Output?

```
map<int,int> m;
m[1]=10;
m[2]=20;
cout << m.size();
```

A. 1
B. 2
C. 3
D. Error

✅ **Answer:** B

---

16. RTTI stands for

A. Run-Time Type Inheritance
B. Run-Time Type Information
C. Real-Time Type Info
D. Runtime Template Info

✅ **Answer:** B

---

17. RTTI is mainly used with

A. Templates
B. Virtual functions
C. Static functions
D. Inline functions

✅ **Answer:** B

---

18. Which operator is used for RTTI?

A. sizeof
B. typeid
C. cast
D. info

✅ **Answer:** B

---

19. Output?

```
int x;
cout << typeid(x).name();
```

A. int
B. i
C. Depends on compiler
D. Error

✅ **Answer:** C

---

20. dynamic_cast works only with

A. Non-polymorphic classes
B. Polymorphic classes
C. Templates
D. STL containers

✅ **Answer:** B

🟡 MEDIUM MCQs (21–40)

---

21. Output?

```
vector<int> v = {10,20,30};
cout << v[1];
```

A. 10
B. 20
C. 30
D. Error

✅ **Answer:** B

---

22. Output?

```
vector<int> v;
v.push_back(1);
v.push_back(2);
v.pop_back();
cout << v.size();
```

A. 1
B. 2
C. 0
D. Error

✅ **Answer:** A

---

23. Which container does NOT allow iteration?

A. vector
B. map
C. stack
D. deque

✅ **Answer:** C

---

24. Output?

```
map<int,string> m;
m[1]="A";
m[1]="B";
cout << m.size();
```

A. 1
B. 2
C. Error
D. Garbage

✅ **Answer:** A

---

25. Which STL container allows duplicate keys?

A. map
B. set
C. multimap
D. unordered_map

✅ **Answer:** C

---

26. Output?

```
queue<int> q;
q.push(10);
q.push(20);
q.pop();
cout << q.front();
```

A. 10
B. 20
C. Error
D. Garbage

✅ **Answer:** B

---

27. Which container is best for stack behavior?

A. vector
B. deque
C. stack
D. list

✅ **Answer:** C

---

28. Output?

```
vector<int> v = {1,2,3};
v.insert(v.begin(), 0);
cout << v[0];
```

A. 1
B. 0
C. Error
D. Garbage

✅ **Answer:** B

---

29. typeid requires which header?

A. <iostream>
B. <typeinfo>
C. <memory>
D. <algorithm>

✅ **Answer:** B

---

30. Output?

```
class A { virtual void f(){} };
A a;
cout << typeid(a).name();
```

A. A
B. class A
C. Compiler dependent
D. Error

✅ **Answer:** C

---

31. Which cast uses RTTI?

A. static_cast
B. const_cast

C. dynamic_cast
D. reinterpret_cast

✅ **Answer:** C

---

32. Output?

```
class Base { virtual void f(){} };
class Derived: public Base {};
Base* b = new Derived();
Derived* d = dynamic_cast<Derived*>(b);
cout << (d != nullptr);
```

A. 0
B. 1
C. Error
D. Garbage

✅ **Answer:** B

---

33. Which container provides random access iterator?

A. vector
B. map
C. list
D. stack

✅ **Answer:** A

---

34. Output?

```
map<int,int> m;
cout << m.empty();
```

A. true
B. false
C. Error
D. Garbage

✅ **Answer:** A

---

35. Which STL container is ordered by default?

A. vector
B. queue
C. map
D. stack

✅ **Answer:** C

---

36. Output?

```
vector<int> v(3,5);
cout << v[2];
```

A. 0
B. 3
C. 5
D. Error

✅ **Answer:** C

---

37. Which function removes all elements from vector?

A. delete
B. erase
C. clear
D. remove

✅ **Answer:** C

---

38. dynamic_cast returns

A. Exception
B. nullptr on failure
C. Garbage
D. Compile error

✅ **Answer:** B

---

39. Which STL container does NOT support push_back()?

A. vector
B. deque
C. list
D. map

✅ **Answer:** D

---

40. RTTI works only if

A. Class has constructor
B. Class has destructor
C. Class is polymorphic
D. Class has data

✅ **Answer:** C

🔴 HARD & TRICKY MCQs (41–60)

---

41. Output?

```
vector<int> v = {1,2,3};
cout << *(v.begin()+1);
```

A. 1
B. 2
C. 3
D. Error

✅ **Answer:** B

---

42. Output?

```
map<int,int> m;
m.insert({2,20});
m.insert({1,10});
for(auto p: m)
```

```
 cout << p.first;
```

A. 21
B. 12
C. 12 (sorted)
D. Random

✅ **Answer:** C

---

43. Output?

```
stack<int> s;
s.push(1);
s.push(2);
s.pop();
cout << s.top();
```

A. 1
B. 2
C. Error
D. Garbage

✅ **Answer:** A

---

44. Which STL container invalidates iterators on insert?

A. vector
B. map
C. set
D. stack

✅ **Answer:** A

---

45. Output?

```
class A { virtual void f(){} };
class B : public A {};
A* a = new A();
B* b = dynamic_cast<B*>(a);
```

```
cout << (b == nullptr);
```

A. 0
B. 1
C. Error
D. Garbage

✅ **Answer:** B

---

46. Which cast is fastest but unsafe?

A. dynamic_cast
B. static_cast
C. const_cast
D. reinterpret_cast

✅ **Answer:** D

---

47. Output?

```
vector<int> v = {1,2,3};
v.erase(v.begin()+1);
cout << v.size();
```

A. 1
B. 2
C. 3
D. Error

✅ **Answer:** B

---

48. Which container is best for key lookup?

A. vector
B. list
C. map
D. stack

✅ **Answer:** C

---

49. Output?

```
typeid(10) == typeid(20)
```

A. true
B. false
C. Error
D. Garbage

✅ **Answer:** A

---

50. Which container has no iterators?

A. stack
B. queue
C. priority_queue
D. All

✅ **Answer:** D

---

51. Output?

```
vector<int> v;
cout << v.capacity();
```

A. 0
B. >=0 (implementation dependent)
C. Error
D. Garbage

✅ **Answer:** B

---

52. RTTI is useful mainly for

A. Templates
B. Runtime type checking
C. Compile-time optimization
D. File handling

✅ **Answer:** B

---

53. Which STL container uses balanced BST internally?

A. vector
B. map
C. queue
D. stack

✅ **Answer:** B

---

54. Output?

```
map<int,int> m;
m[5]=50;
cout << m[5];
```

A. 5
B. 50
C. Error
D. Garbage

✅ **Answer:** B

---

55. Which container allows duplicate values but no keys?

A. set
B. multiset
C. map
D. vector

✅ **Answer:** B

---

56. dynamic_cast requires

A. Base pointer
B. Virtual function
C. RTTI enabled
D. All

✅ **Answer:** D

---

57. Output?

```
vector<int> v = {1,2,3};
cout << v.at(3);
```

A. 0
B. Garbage
C. Exception
D. Error

✅ **Answer:** C *(out_of_range)*

---

58. Which STL container has O(1) access time?

A. vector
B. list
C. map
D. stack

✅ **Answer:** A

---

59. typeid on polymorphic object gives

A. Static type
B. Dynamic type
C. Base type
D. Compile error

✅ **Answer:** B

---

60. Best practice

A. Prefer dynamic_cast everywhere
B. Avoid RTTI when possible
C. Use reinterpret_cast
D. Disable STL

✅ **Answer:** B