1 What is an array in C++?	
A A collection of objects	B A collection of functions
C A collection of data of the same type	D A collection of random data
Answer: C	

2 What does a pointer store?	
A Memory address	B A value
C A reference	D An integer
Answer: A	

3 What is a function prototype used for?	
A To define a new data type	B To declare a variable
C To declare a function	D To define a function
Answer: C	

4 Which keyword is used for dynamic memory allocation?	
A alloc	B new
C malloc	D alloc_ptr
Answer: B	

5 What is recursion in programming?	
A A function that calls itself	B A function that returns no value
C A function with a loop	D A function with a switch statement
Answer: A	

6 What is a class in C++?	
A A blueprint for creating objects	B A data type in C++
C A function in C++	D An array in C++
Answer: A	

7 Which access specifier allows access from outside the class?	
A public	B private
C protected	D default
Answer: A	

8 What is a constructor in C++?	
A A member function of a class	B A special function for memory allocation
C A function with no parameters	D A destructor for freeing memory
Answer: D	

9 Which data structure represents a last-in, first-out (LIFO) order?	
A Queue	B Stack
C Linked List	D Array
Answer: B	

10 What is the purpose of the new operator in C++?	
A To declare a new variable	B To allocate memory for an object
C To define a new function	D To create a new class
Answer: B	

11 What is a lambda function in C++?	
A A function that takes no arguments	B A function that can only be used with main()
C An anonymous function that can capture variables from its surrounding scope	D A function that cannot return a value
Answer: C	

12 In C++, what is the purpose of the typeid operator?	
A To check if a variable is of a specific type	B To convert between data types
C To find the memory address of a variable	D To perform bitwise operations
Answer: A	

13 What is the purpose of the const member function in a class?	
A To declare a constant variable in the class	B To define a constructor for the class
C To indicate that the member function does not modify the object's data members	D To create a static member function
Answer: C	

14 What is a virtual function in C++ and when is it used?	
A A function that is declared as virtual in a base class and can be overridden in derived classes	B A function that can only be accessed within the class where it is defined
C A function that is declared as static in a class	D A function that takes no arguments
Answer: A	

15 What is the difference between a reference and a pointer in C++?	
A References are used for dynamic memory allocation, while pointers are used for static memory allocation.	B References are used for indirect access to variables, while pointers are used for direct access.
C References are automatically dereferenced, while pointers require explicit dereferencing.	D References can be reassigned to point to different objects, while pointers cannot.
Answer: C	

16 What is the purpose of the break statement in C++?	
A To exit the program	B To exit a loop or switch statement prematurely
C To continue to the next iteration of a loop	D To skip a specific case in a switch statement
Answer: B	

17 How do you declare a constant variable in C++?	
A Using the const keyword before the variable type	B Using the constant keyword before the variable name
C Using the static keyword before the variable name	D Using the final keyword before the variable type
Answer: A	

18 What is the purpose of the cin object in C++?	
A To output data to the console	B To input data from the console
C To perform mathematical calculations	D To declare variables
Answer: B	

19 Which header file should be included to use the cin and cout objects in C++?	
A <stdio.h>	B <iostream>
C <stdlih>	D <math.h>
Answer: B	

20 What is the difference between = and == in C++?

A = is used for assignment, while == is used for comparison.	B = is used for comparison, while == is used for assignment.
C Both = and == are used for assignment.	D Both = and == are used for comparison.
Answer: A	

21 What is the purpose of the continue statement in C++?	
A To exit a loop prematurely	B To break out of a switch statement
C To skip the rest of the code in a loop iteration and move to the next iteration	D To pause the program execution
Answer: C	

22 Which operator is used to access the address of a variable in C++?	
A &	B *
C ->	D .
Answer: A	

23 What does the nullptr keyword represent in C++?	
A A null pointer	B A pointer to an undefined memory location
C A pointer to the current object	D A pointer to a function
Answer: A	

24 What is the purpose of the static keyword in C++?	
A To declare a constant variable	B To declare a global variable
C To specify that a variable is shared among all instances of a class	D To indicate that a variable is temporary
Answer: C	

25 What is an array in C++?	
A A collection of objects	B A collection of functions
C A collection of data of the same type	D A collection of random data
Answer: C	

26 What is the purpose of the override keyword in C++?	
A To indicate that a function is a constructor	B To specify the base class of an object

C To explicitly indicate that a function is intended to override a virtual function in a base class	D To create a static member function
Answer: C	

27 In C++, what is the order of destruction of objects when they go out of scope?	
A From child to parent classes	B In the order they were created
C From parent to child classes	D In reverse order of creation
Answer: C	

28 What is a template class in C++?	
A A class that can only be used with a single data type	B A class that can be instantiated with different data types
C A class that is defined in a template file	D A class that is declared as static
Answer: B	

29 In C++, what is the purpose of the nothrow keyword when used with the new operator?	
A To allocate memory without initializing it	B To allocate memory and initialize it with zeros
C To allocate memory and throw an exception on failure	D To allocate memory and return a null pointer on failure
Answer: D	

30 What is the purpose of the const_cast operator in C++?	
A To perform a dynamic cast between class objects	B To cast a constant object to a non-constant object
C To cast a non-constant object to a constant object	D To cast between unrelated data types
Answer: B	

31 What is the purpose of the default case in a switch statement?	
A To indicate the end of the switch statement	B To specify the first case to execute
C To provide a default value if no case matches	D To skip the switch statement
Answer: C	

32 What is the purpose of the while loop in C++?

A To execute a block of code a specified number of times	B To execute a block of code repeatedly as long as a condition is true
C To execute a block of code once and then exit	D To execute a block of code in parallel
Answer: B	

33 What is the purpose of the break statement in a loop?	
A To continue to the next iteration of the loop	B To exit the program
C To exit the loop prematurely	D To skip a specific case in a switch statement
Answer: C	

34 Which type of memory allocation allows you to allocate memory during program execution?	
A Static memory allocation	B Dynamic memory allocation
C Automatic memory allocation	D Manual memory allocation
Answer: B	

35 What is the scope of a local variable in C++?	
A Limited to the function where it is declared	B Limited to the class where it is declared
C Limited to the file where it is declared	D Limited to the block where it is declared
Answer: D	

36 What is the purpose of the else statement in an if condition?	
A To specify the condition	B To indicate the end of the if block
C To provide an alternative code block to execute if the if condition is false	D To repeat the if condition
Answer: C	

37 Which operator is used to access the elements of an array in C++?	
A .	B ->
C []	D ()
Answer: C	

38 What is the purpose of the goto statement in C++?	
A To exit a loop prematurely	B To jump to a labeled statement within the same function

C To call a function	D To repeat a code block
Answer: B	

39 What is the purpose of the continue statement in C++?	
A To exit a loop prematurely	B To break out of a switch statement
C To skip the rest of the code in a loop iteration and move to the next iteration	D To pause the program execution
Answer: C	

40 Which operator is used to access the address of a variable in C++?	
A &	B *
C ->	D .
Answer: A	

41 What is the purpose of the static_cast operator in C++?	
A To perform a dynamic cast between class objects	B To cast between unrelated data types
C To cast a pointer to an object of a related class	D To cast a constant object to a non-constant object
Answer: B	

42 What is the RAII (Resource Acquisition Is Initialization) principle in C++?	
A It is a design pattern that ensures all resources are released in a program.	B It is a type of smart pointer in C++.
C It is a keyword used for defining recursive functions.	D It is a type of data structure in C++.
Answer: A	

43 In C++, what is a friend function?	
A A function that can access private and protected members of a class	B A function that is a member of a class
C A function that is declared as static	D A function that cannot access any class members
Answer: A	

44 What is the purpose of the std::move function in C++?	
A To create a new object	B To move the content of one object to another efficiently

C To copy the content of one object to another	D To allocate memory for an object
Answer: B	

45 What is the Big O notation used for in algorithm analysis?	
A To represent the number of bytes used by an algorithm	B To represent the number of comparisons in an algorithm
C To represent the worst-case time complexity of an algorithm	D To represent the best-case time complexity of an algorithm
Answer: C	

46 What is the correct syntax to declare a class in C++?	
A class MyClass{};	B class MyClass()
C MYClass()	D MyClass class{};
Answer: A	

47 In C++, an object is:	
A An instance of a class	B Method in Class
C Variable in Class	D None
Answer: A	

48 Which access specifier allows members to be accessed by any code in the program?	
A private	B protected
C global	D public
Answer: D	

49 A constructor in C++:	
A Is automatically called when an object is created	B Is used to destroy objects
C Can be inherited from a base class	D It is used to allocate memory for class
Answer: A	

50 What is the purpose of a destructor in C++?	
A To deallocate memory and perform cleanup before an object is destroyed	B To initialize class members
C To delete object	D reinitialize object
Answer: A	

51 Which of the following feature of OOPs is not used in the following C++ code? class A{
```
   int i;
   public:
   void print() {cout << "hello" << i;}
}
class B : public A{
   int j;
   public:
   void assign (int a ) {k = a;}
}
```

A Abstraction	B Encapsulation
C Polymorphism	D Inheritance
Answer: C	

52 Using friend operator function, following perfect set of operators may not be overloaded.	
A = , ( ) , [ ] , ->	B <<, = = , [ ] , >>
C <<, / = , [ ] , >>	D <<, =*= , [ ] , >>
Answer: A	

53 When does the copy constructor get called in C++?	
A When an object is passed to a function by value	B When an object is returned from a function by value
C When an object is explicitly destroyed using the delete keyword	D When an object is created using another object of the same class
Answer: D	

54 What is the output of the following code? #include <iostream>
```
using namespace std;

class Base {
public:
   Base() { cout << "Base constructor" << endl; }
   virtual ~Base() { cout << "Base destructor" << endl; }
};

class Derived : public Base {
public:
   Derived() { cout << "Derived constructor" << endl; }
   ~Derived() { cout << "Derived destructor" << endl; }
};
```

```
int main() {
   Base* ptr = new Derived();
   delete ptr;
   return 0;
}
```

A Base constructor, Derived constructor, Derived destructor, Base destructor	B Derived constructor, Base constructor, Base destructor, Derived destructor
C Base constructor, Derived constructor, Base destructor	D Derived constructor, Base destructor
Answer: A	

55 What is the correct syntax of accessing a static member of a Class? class A

```
{
 public:
  static int value;
}
```

A A.value	B A::value
C A->value	D None
Answer: B	

56 Which operator should be overloaded in the following code to make the program error free? #include <iostream>

```
#include <string>
using namespace std;
class Box{
 int capacity;
public:
 Box(){}
 Box(double capacity){
  this->capacity = capacity;
 }
};
int main(int argc, char const *argv[])
{
 Box b1(10);
 Box b2 = Box(14);
 if(b1 == b2){
  cout<<"Equal";
 }
 else{
  cout<<"Not Equal";
 }
 return 0;
}
```

A +	B =
C =='	D ()
Answer: C	

57 When base class is derived in protected mode, then_____ . 1. public members of base class become private members of derived class. 2. public members of base class become protected members of derived class. 3. public members of base class become public members of derived class. 4. protected members of base class become protected members of derived class. 5. protected members of base class become private members of derived class. 6. protected members of base class become public members of derived class.

A Only 2,4	B Only 3,6
C Only 4,5	D Only 1
Answer: A	

58 Give the function prototype of the operator function which we need to define in this program so that the program has no errors #include <iostream>

```
#include <string>
using namespace std;
class Box{
 int capacity;
public:
 Box(){}
 Box(double capacity){
  this->capacity = capacity;
 }
};

int main(int argc, char const *argv[])
{
 Box b1(10);
 Box b2 = Box(14);
 if(b1 == b2){
  cout<<"Equal";
 }
 else{
  cout<<"Not Equal";
 }
 return 0;
}
```

a)

A bool operator==(Box b);	B Box operator==();

C bool operator==(Box b){}	D Box operator==();
Answer: A	

59 In case of inheritance where both base and derived class are having constructor and destructor, then which if the following are true ? 1. Constructors are executed in their order of derivation 2. Constructors are executed in reverse order of derivation 3. Destructors are executed in their order of derivation 4. Destructors are executed in reverse order of derivation	
A Only 1,4	B Only 3,2
C Only 4,3	D None
Answer: A	

60 What is the difference between an abstract class and an interface in C++?	
A An abstract class cannot have constructors, while an interface can.	B An abstract class can have data members, while an interface cannot.
C An abstract class can have both concrete and pure virtual functions, while an interface can only have pure virtual functions.	D They reduce the amount of Memory uses when use interface
Answer: C	

61 Which of the following statements about multidimensional arrays in C++ is correct?	
A Multidimensional arrays must have the same number of elements in each dimension.	B Multidimensional arrays are implemented as arrays of pointers to arrays.
C Multidimensional arrays can have different data types in each dimension	D Multidimensional arrays can only have two dimensions
Answer: B	

62 Given the following code snippet:

```
int x = 10;
int *ptr1 = &x;
int *ptr2 = ptr1;
```

What does ptr2 now point to?

A A different memory location than ptr1.	B The address of x.
C The value of x.	D Null pointer.
Answer: B	

63 In C++, what is the difference between pass by value and pass by reference?	
A Pass by value modifies the original argument, while pass by reference does not.	B Pass by reference creates a new copy of the argument, while pass by value does not
C Pass by value passes the memory address of the argument, while pass by reference passes the actual value.	D Pass by reference allows the function to modify the original argument, while pass by value does not.
Answer: D	

64 What is the scope of a variable declared inside a function in C++?	
A The variable is only accessible within the function where it is declared.	B The variable is accessible from any function within the same file.
C The variable is accessible from any function in the program	D The variable is accessible only from the main function.
Answer: A	

65 What is the purpose of the delete operator in C++?	
A To release memory allocated with new.	B To deallocate stack memory.
C To remove a file from the filesystem	D To free memory allocated with malloc.
Answer: A	

66 In C++, when is it necessary to use dynamic memory allocation instead of stack memory	
A Dynamic memory is always preferred over stack memory.	B When the size of the data is known at compile time.
C When the size of the data is known at runtime.	D Dynamic memory should never be used.
Answer: C	

67 Using friend operator function, following perfect set of operators may not be overloaded.	
A = , ( ) , [ ] , ->	B <<, = = , [ ] , >>
C <<, / = , [ ] , >>	D <<, =*= , [ ] , >>
Answer: A	

68 What is "method overloading" in OOP?	
A A technique to overload a class with too many methods.	B The process of providing multiple definitions for the same method in a class.
C A way to hide methods in a class	D A way to create new methods in a subclass.
Answer: B	

69 How is a static member variable different from an instance member variable in a class?	
A Static member variables are not accessible from outside the class.	B Static member variables have the same value for all instances of the class.
C Static member variables can only be modified within the constructor.	D Static member variables are always initialized to zero.
Answer: B	

70 When you pass an object to a C++ function by value, what happens?	
A The original object is modified.	B A copy of the object is created within the function.
C The function cannot access the object's members.	D The object is deleted.
Answer: B	

71 Output-based Question on Static Members and Objects:

```
#include <iostream>


class MyClass {
public:
```

```
    static int count;
    MyClass() {
        count++;
    }
};


int MyClass::count = 0;


int main() {
    MyClass obj1, obj2, obj3;
    std::cout << MyClass::count << std::endl;
    return 0;
}
```

What is the output of this C++ program?

A 3	B 0
C 1	D Compiler Error
Answer: C	

72 When an object is created in C++, where is its memory allocated by default?	
A On the stack.	B On the heap.
C In global memory.	D In CPU registers.
Answer: A	

73 Output-based Question on Array Manipulation:

```
#include <iostream>


void modifyArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] *= 2;
    }
}
```

```
int main() {
   int arr[] = {1, 2, 3, 4, 5};
   modifyArray(arr, 5);
   for (int i = 0; i < 5; i++) {
      std::cout << arr[i] << " ";
   }
   return 0;
}
```

What is the output of this C++ program?

A 2 4 6 8 10	B 1 2 3 4 5
C 0 0 0 0 0	D 1 4 9 16 25
Answer: A	

74 What is a "dangling pointer" in C++?	
A A pointer that is uninitialized.	B A pointer that has been deallocated but still points to a memory location.
C A pointer that points to a valid object.	D A pointer with a value of 0.
Answer: B	

75 Output-based Question on Access Specifiers:

```
#include <iostream>

class MyClass {
private:
   int x;
public:
   int y;
protected:
   int z;
};

int main() {
   MyClass obj;
```

```
    obj.y = 10;
    obj.z = 20;
    std::cout << obj.y << " " << obj.z << std::endl;
    return 0;
}
```

What is the output of this C++ program?

A 10 20	B Compiler Error
C 10 0	D 0 20
Answer: B	

76 What is the purpose of a function prototype in C++?	
A To define the function's implementation.	B To declare the function's name and return type.
C To specify the function's arguments.	D To provide the function's documentation.
Answer: B	

77 What is the purpose of the const keyword in the following declaration?

```
const int *ptr;
```

A It makes ptr a constant pointer.	B It makes the integer pointed to by ptr constant.
C It prevents ptr from being assigned to another pointer.	D It specifies that ptr is a pointer to a constant integer.
Answer: D	

78 What is the output of the following code snippet?

```
int arr[] = {1, 2, 3, 4, 5};
int *ptr = arr;
cout << *ptr << endl;
```

A 1	B 2
C 3	D 4
Answer: A	

79 What is function overloading in C++?	
A A feature that allows a function to have multiple implementations with different names.	B A feature that allows a function to have multiple implementations with the same name but different parameters.
C A feature that allows a function to be called from multiple places in the code	D A feature that prevents a function from being overloaded
Answer: B	

80 What does the arrow operator (->) do in C++?	
A It performs bitwise operations	B It dereferences a pointer to access a member of a structure or class.
C It increments a pointer by one	D It checks if a pointer is valid.
Answer: B	

81 What is tail recursion?	
A A type of recursion that involves calling multiple functions.	B A type of recursion where the recursive call is the last action in the function.
C A type of recursion that uses a loop instead of recursive calls	D A type of recursion that does not have a base case
Answer: B	

82 Which OOP feature promotes code reusability by allowing one class to inherit the properties and behaviors of another class?	
A Polymorphism	B Encapsulation
C Inheritance	D Abstraction
Answer: C	

83 In C++, what is the difference between a class member declared as public and one declared as private?

A public members can only be accessed within the class, while private members can be accessed from anywhere.	B public members can be accessed from anywhere, while private members can only be accessed within the class.
C public members have read-only access, while private members have read and write access.	D There is no difference; both can be accessed from anywhere.
Answer: B	

84 In C++, which access specifier allows class members to be accessed from derived classes?	
A public	B private
C protected	D Internal
Answer: B	

85 What is a "constructor" in a C++ class?	
A A member function that destroys an object.	B A special member function that initializes objects of the class
C A function that defines the interface of a class.	D A function that cannot have parameters.
Answer: B	

86 Output-based Question on Access Specifiers:

```
#include <iostream>


class MyClass {
private:
   int x;
public:
   int y;
protected:
   int z;
};
```

```
int main() {
   MyClass obj;
   obj.y = 10;
   obj.z = 20;
   std::cout << obj.y << " " << obj.z << std::endl;
   return 0;
}
```

What is the output of this C++ program?

A 10 20	B Compiler Error
C 10 0	D 0 20
Answer: B	

```
87 #include <iostream>

class MyArray {
private:
   int arr[5];
public:
   MyArray() {
      for (int i = 0; i < 5; i++) {
         arr[i] = 0;
      }
   }
   void set(int index, int value) {
      if (index >= 0 && index < 5) {
         arr[index] = value;
      }
   }
   int get(int index) {
      if (index >= 0 && index < 5) {
         return arr[index];
      }
      return -1; // Invalid index
   }
   void display() {
      for (int i = 0; i < 5; i++) {
         std::cout << arr[i] << " ";
      }
      std::cout << std::endl;
   }
};
```

```
int main() {
   MyArray obj;
   obj.set(2, 42);
   obj.set(4, 99);
   std::cout << obj.get(2) << " " << obj.get(4) << std::endl;
   obj.display();
   return 0;
}
```

A 42 99 followed by 0 0 42 0 99	B 0 0 followed by 0 0 0 0 0
C 0 0 followed by 0 0 42 0 99	D Compiler Error
Answer: C	

```
88 #include <iostream>

class Shape {
public:
   virtual void draw() {
      std::cout << "Drawing a shape" << std::endl;
   }
};

class Circle : public Shape {
public:
   void draw() override {
      std::cout << "Drawing a circle" << std::endl;
   }
};

class Rectangle : public Shape {
public:
   void draw() override {
      std::cout << "Drawing a rectangle" << std::endl;
   }
};

int main() {
   Shape* shapePtr;
   Circle circle;
   Rectangle rectangle;

   shapePtr = &circle;
   shapePtr->draw();

   shapePtr = &rectangle;
```

```
    shapePtr->draw();

    return 0;
}
```

What is the output of this C++ program?

A Drawing a shape Drawing a shape	B Drawing a circle Drawing a shape
C Drawing a shape Drawing a rectangle	D Drawing a circle Drawing a rectangle

Answer: D

```
89 #include <iostream>

class MyClass {
public:
    MyClass() {
        std::cout << "Constructor called" << std::endl;
    }
    ~MyClass() {
        std::cout << "Destructor called" << std::endl;
    }
};

int main() {
    MyClass obj1;
    {
        MyClass obj2;
    }
    MyClass obj3;
    return 0;
}
```

A Constructor called Constructor called Destructor called Destructor called Constructor called Destructor called	B Constructor called Constructor called Destructor called Destructor called Constructor called Destructor called
C Constructor called Constructor called Constructor called Destructor called Destructor called	D Constructor called Destructor called Constructor called Destructor called Constructor called

Destructor called	Destructor called
Answer: B	

90 #include <iostream>

```
int main() {
   int num = 42;
   int* ptr1 = &num;
   int** ptr2 = &ptr1;

   std::cout << num << " ";
   std::cout << *ptr1 << " ";
   std::cout << **ptr2 << std::endl;

   return 0;
}
```

A 42 42 42	B 42 0 0
C 42 42 0	D 42 0 42
Answer: A	