C++ variables ,literals and Constants

Question 1:

What is a variable in programming?

a) A function
b) A container to hold data
c) A loop

d) A constant

Answer: b) A container to hold data

Question 2:

Which data type is used for variables that can only hold integers?

a) double
b) char
c) int

d) float

Answer: c) int

Question 3:

What is the preferred practice for naming variables?

a) Using uppercase characters
b) Starting with a number
c) Starting with a lowercase character

d) Using special characters

Answer: c) Starting with a lowercase character

Question 4:

Which of the following is a valid rule for naming variables?

a) A variable name can only have alphabets and numbers.
b) A variable name cannot begin with a number.
c) A variable name must start with an underscore.

d) A variable name can be a keyword.

Answer: b) A variable name cannot begin with a number.

Question 5:

What is a literal in programming?

a) A comment in the code
b) Data used for representing fixed values
c) A function

d) A variable

Answer: b) Data used for representing fixed values

Question 6:

Which type of literal represents numeric values without any fractional or exponential part?

a) Floating-point literals
b) Character literals
c) Integer literals

d) String literals

Answer: c) Integer literals

Question 7:

What is the escape sequence for a newline character?

a) \n
b) \t
c) \b
d) \r

Answer: a) \n

Question 8:

How do you create a constant in C++?

a) Using the keyword "variable"
b) Using the keyword "const"
c) Using the keyword "constant"

d) Using the keyword "let"

Answer: b) Using the keyword "const"

Question 9:

Which of the following is a valid floating-point literal?

a) 10
b) 3.14E2
c) 'a'
d) 0xFF

Answer: b) 3.14E2

Question 10:

What is the purpose of escape sequences in C++?

a) To create new variables
b) To represent fixed values
c) To escape from loops
d) To use characters that cannot be typed directly or have special meanings

Answer: d) To use characters that cannot be typed directly or have special meanings

DATA TYPES IN C++

Question 11:

What is the size of the character type in C++?

a) 8 bits
b) 16 bits
c) 32 bits

d) 64 bits

Answer: a) 8 bits

Question 12:

Which type represents the most natural size of an integer for the machine?

a) char
b) int
c) float

d) double

Answer: b) int

Question 13:

What is the purpose of the void type in C++?

a) It represents a single-precision floating-point value.
b) It represents the absence of a type.
c) It is used for boolean values.

d) It is a double-precision floating-point value.

Answer: b) It represents the absence of a type.

Question 14:

What is the type of the variable "cond" in the given code?

```
bool cond = false;
```

a) int
b) char
c) bool

d) float

Answer: c) bool

Question 15:

Which type represents a double-precision floating-point value in C++?

a) float
b) double
c) int

d) char

Answer: b) double

Question 16:

What is the type of the variable "variable_name" in the following code?

cpp
Copy code
int variable_name;
a) char
b) int
c) float
d) double

Answer: b) int

Question 17:
Which type represents a single-precision floating-point value in C++?

a) double
b) float
c) int
d) char

Answer: b) float

Question 18:
What is the purpose of the boolean type in C++?

a) To represent floating-point values
b) To represent integers
c) To represent the absence of a type
d) To represent true or false values

Answer: d) To represent true or false values

Question 19:
What does the void type signify in C++?

a) A single-precision floating-point value
b) The absence of a type
c) A double-precision floating-point value
d) An integer value

Answer: b) The absence of a type

Question 20:
If a variable is declared with the type "char," how many bytes does it typically occupy in memory?

a) 2 bytes
b) 4 bytes
c) 8 bytes
d) 1 byte

Answer: d) 1 byte

LOOPS IN C++

Question 21:
Which part of the for loop is responsible for initializing the loop control variable?

a) Initialization
b) Condition
c) Increment/Decrement
d) Code block

Answer: a) Initialization

Question 22:
When is the condition in a for loop checked?

a) After every iteration
b) Before the loop starts
c) During initialization

d) After the increment/decrement operation

Answer: a) After every iteration

Question 23:
In a while loop, when is the condition checked?

a) After every iteration
b) Before the loop starts
c) During initialization
d) After the increment/decrement operation

Answer: b) Before the loop starts

Question 24:
What does the do-while loop guarantee regarding code execution?

a) The code will execute only once.
b) The code will execute multiple times.
c) The code will execute based on the condition.
d) The code will never execute.

Answer: a) The code will execute at least once.

Question 25:
In a do-while loop, when is the condition checked?

a) After every iteration
b) Before the loop starts
c) During initialization
d) After the code block is executed

Answer: a) After every iteration

Question 26:
Which conditional statement is used to execute a block of code if a specified condition evaluates to true?

a) else
b) if-else
c) switch
d) if

Answer: d) if

Question 27:
In the if-else statement, what happens if the condition is false?

a) The program terminates.
b) The code block is skipped.
c) The else block is executed.
d) An error occurs.

Answer: c) The else block is executed.

Question 28:
What is the purpose of the else if part in the if-else if-else statement?

a) It is used for loop control.
b) It specifies the default case.
c) It tests additional conditions sequentially.
d) It is optional and can be omitted.

Answer: c) It tests additional conditions sequentially.

Question 29:
In the switch statement, what is the role of the break statement?

a) It starts the execution of the switch block.
b) It skips the current case and moves to the next one.
c) It exits the switch block after executing a case.
d) It is used to declare variables.

Answer: c) It exits the switch block after executing a case.

Question 30:
What is the purpose of the default case in a switch statement?

a) It specifies the first case to be executed.
b) It is required in every switch statement.
c) It is executed if none of the case values match the expression.
d) It defines the default value of the switch variable.

Answer: c) It is executed if none of the case values match the expression.

Question 31:
Which of the following statements is used to execute a block of code when a condition is false?

a) else
b) if
c) switch
d) if-else

Answer: a) else

Question 32:
In the if-else if-else statement, what happens if multiple conditions are true?

a) All code blocks associated with true conditions are executed.
b) Only the first true condition's code block is executed.
c) The conditions are ignored, and the default block is executed.
d) An error occurs.

Answer: b) Only the first true condition's code block is executed.

Question 33:
What is the primary purpose of the switch statement?

a) To execute a block of code repeatedly.
b) To define a loop.
c) To select one of many code blocks to be executed.
d) To declare variables.

Answer: c) To select one of many code blocks to be executed.

Question 34:
In the switch statement, what happens if there is no break statement after a case?

a) It causes a compilation error.
b) It has no effect on the program's execution.
c) It skips the current case and moves to the next one.
d) It results in a runtime error.

Answer: c) It skips the current case and moves to the next one.

Question 35:
Which statement is used to exit the current iteration of a loop prematurely?

a) exit
b) return
c) break
d) continue

Answer: c) break

Question 36:
What is the purpose of the continue statement in a loop?

a) To exit the loop.
b) To skip the remaining code in the loop and move to the next iteration.
c) To restart the loop from the beginning.
d) To jump to a specific label in the program.

Answer: b) To skip the remaining code in the loop and move to the next iteration.

Question 37:
Which statement is used to skip the rest of the code in the current iteration and start the next iteration of a loop?

a) return
b) break
c) continue
d) exit

Answer: c) continue

Question 38:
In a for loop, where is the initialization part typically located?

a) At the end of the loop body
b) Before the loop body
c) After the loop condition
d) Inside the loop condition

Answer: b) Before the loop body

Question 39:
What is the role of the default case in a switch statement?

a) It specifies the first case to be executed.
b) It is executed if none of the case values match the expression.
c) It is optional and can be omitted.
d) It defines the default value of the switch variable.

Answer: b) It is executed if none of the case values match the expression.

Question 40:
In the if-else statement, what happens if there is no else block?

a) It causes a compilation error.
b) It is optional and can be omitted.
c) The program terminates.
d) The if block is skipped if the condition is false.

Answer: b) It is optional and can be omitted.

C++ Type Conversion

Question 41:
What is type conversion in C++?

a) Assigning a value to a variable
b) Printing the value of a variable
c) Converting data of one type to another
d) Declaring a variable

Answer: c) Converting data of one type to another

Question 42:
Which type of conversion is done automatically by the compiler?

a) Explicit conversion
b) User-defined conversion
c) Implicit conversion
d) Function-style conversion

Answer: c) Implicit conversion

Question 43:
In the given code, what type of conversion is demonstrated?

```
int num_int = 9;
double num_double = num_int;
```

a) Explicit conversion
b) Implicit conversion
c) C-style type casting
d) Function notation

Answer: b) Implicit conversion

Question 44:
What is the output of the following code?

```
int num_int = 9;
double num_double = num_int;
cout << "num_int = " << num_int << endl;
cout << "num_double = " << num_double << endl;
```

a) num_int = 9, num_double = 9
b) num_int = 9, num_double = 9.0
c) num_int = 9.0, num_double = 9
d) num_int = 9.0, num_double = 9.0

Answer: b) num_int = 9, num_double = 9.0

Question 45:
What is the term for manually changing data from one type to another in C++?

a) Implicit conversion
b) Explicit conversion
c) Type declaration
d) Function notation

Answer: b) Explicit conversion

Question 46:
Which type of conversion is also known as type casting?

a) Implicit conversion
b) Explicit conversion
c) C-style type casting
d) Function notation

Answer: b) Explicit conversion

Question 47:
What are the two major ways of performing explicit conversion in C++?

a) Implicit casting and explicit casting
b) Type declaration and type conversion
c) C-style type casting and function notation
d) Auto casting and manual casting

Answer: c) C-style type casting and function notation

Question 48:
In C++, what is the purpose of C-style type casting?

a) To perform implicit conversion
b) To perform explicit conversion
c) To declare variables
d) To print the value of a variable

Answer: b) To perform explicit conversion

Question 49:
What is the output of the following code?

```
double num_double = 3.56;
int num_int1 = (int)num_double;
cout << "num_int1   = " << num_int1 << endl;
```

a) num_int1 = 3.56
b) num_int1 = 3
c) num_int1 = 4

d) num_int1 = 3.0

Answer: b) num_int1 = 3

Question 50:
In the given code, what is the alternative way of performing explicit conversion?

```
double num_double = 3.56;
int num_int2 = int(num_double);
```

a) C-style type casting
b) Implicit conversion
c) Function-style conversion
d) Auto conversion

Answer: c) Function-style conversion

Question 56:
What is an array in C++?

a) A collection of pointers
b) A collection of items stored in contiguous memory locations
c) A collection of functions
d) A collection of structures

Answer: b) A collection of items stored in contiguous memory locations

Question 57:
Which of the following is an example of array declaration with a specified size?

```
int arr1[10];
```

a) int arr[10];
b) int arr[ ];
c) int arr[] = {10};
d) int arr[10] = {0};

Answer: a) int arr[10];

Question 58:
What is the purpose of the statement int arr[] = {10, 20, 30, 40};?

a) It declares an array without specifying the size.
b) It initializes an array with the specified elements.
c) It creates an array of size 4 with default values.
d) It declares and initializes a variable.

Answer: b) It initializes an array with the specified elements.

Question 59:
What is a characteristic of arrays in C++?

a) Arrays can only store elements of the same data type.
b) Arrays have a dynamic size.
c) Arrays can only have one dimension.
d) Arrays can store elements of different data types.

Answer: a) Arrays can only store elements of the same data type.

Question 60:
What is the output of the following code snippet?

```
int marks[4] = {23, 45, 56, 89};
int marks_in_english[6];
marks_in_english[0] = 35;
marks_in_english[1] = 76;
marks_in_english[2] = 88;
marks_in_english[3] = 89;
cout << marks_in_english[1] << endl;
```

a) 45
b) 76
c) 88
d) 89

Answer: b) 76

Question 61:
Which statement is correct regarding the limitations of arrays?

a) Arrays have a fixed number of elements decided at runtime.
b) Arrays in C++ are dynamic.
c) Insertion and deletion of elements in arrays are always efficient.
d) Arrays can store elements of different data types.

Answer: a) Arrays have a fixed number of elements decided at runtime.

Question 62:
What is the output of the following program?

```
int arr[4] = {10, 20, 30, 40};
```

```
for(int i = 0; i < 4; i++) {
    cout << arr[i] << " ";
}
```

a) 10 20 30 40
b) 40 30 20 10
c) 0 0 0 0
d) 10 0 20 0

Answer: a) 10 20 30 40

Question 63:
What is the output of the following program?

```
int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
cout << arr[1][2];
```

a) 1
b) 2
c) 4
d) 6

Answer: d) 6

Question 64:
What does the following program do?

```
int arr[3][4];
cout << "Enter the Elements\n";
for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 4; j++) {
        cin >> arr[i][j];
    }
}
```

a) Declares a 2D array without initializing it.
b) Initializes a 2D array with default values.
c) Prints the elements of a 2D array.
d) Takes input for a 2D array from the user.

Answer: d) Takes input for a 2D array from the user.

Question 65:
What is the primary advantage of using a 2D array?

a) It allows dynamic sizing.
b) It provides random access of elements.
c) It stores elements of different data types.
d) It allows efficient insertion and deletion of elements.

Answer: b) It provides random access of elements.

Question 66:
What is the output of the following program?

```
int arr[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
cout << arr[2][1];
```

a) 2
b) 6
c) 10
d) 12

Answer: c) 10

Question 67:
What is the output of the following program?

```
int main()
{
   int i;
   int arr[5] = {1};
   for (i = 0; i < 5; i++)
      printf("%d ", arr[i]);
   return 0;
}
```

(A) 1 followed by four garbage values.

(B) 1 0 0 0 0

(C) 1 1 1 1 1

(D) 0 0 0 0 0

Solution: (B) 1 0 0 0 0

Question 68:
What is the output of the following program?

```
int main()
```

```
{
   int a[][] = {{1,2},{3,4}};
   int i, j;
   for (i = 0; i < 2; i++)
      for (j = 0; j < 2; j++)
         printf("%d ", a[i][j]);
   return 0;
}
```

(A) 1 2 3 4

(B) Compiler Error in line "int a[][] = {{1,2},{3,4}};"

(C) 4 garbage values

(D) 4 3 2 1

Solution: (B) Compiler Error in line "int a[][] = {{1,2},{3,4}};"

Question 69:
Consider the following declaration of a ‘two-dimensional array in C:

```
char a[100][100];
```

Assuming that the main memory is byte-addressable and that the array is stored starting from memory address 0, the address of a[40][50] is: (GATE CS 2002)

(A) 4040

(B) 4050

(C) 5040

(D) 5050

Solution: (B) 4050

Question 70:
For a C program accessing X[i][j][k], the following intermediate code is generated by a compiler. Assume that the size of an integer is 32 bits and the size of a character is 8 bits. (GATE-CS-2014)

```
t0 = i * 1024
t1 = j * 32
t2 = k * 4
t3 = t1 + t0
```

```
t4 = t3 + t2
t5 = X[t4]
```

Which one of the following statements about the source code of C program is correct?

(A) X is declared as "int X[32][32][8]"

(B) X is declared as "int X[4][1024][32]"

(C) X is declared as "char X[4][32][8]"

(D) X is declared as "char X[32][16][2]"

Solution: (A) X is declared as "int X[32][32][8]"

Question 71:
Assume the following C variable declaration:

```
int *A[10], B[10][10];
```

Of the following expressions

I. A[2]

II. A[2][3]

III. B[1]

IV. B[2][3]

which will not give compile-time errors if used as left-hand sides of assignment statements in a C program (GATE CS 2003)?

(A) I, II, and IV only

(B) II, III, and IV only

(C) II and IV only

(D) IV only

Solution: (C) II and IV only

Types of Functions in C++

Question 72:
What is the purpose of the following code snippet?

```
#include <iostream>
#include <math.h>
using namespace std;

// User-defined function for finding factors of a number

void factorise( int num)
{
    for(int i = 1; i <= num; i++)
      if (num % i == 0)
         cout<<i<<" ";
}

int main()
{
  // Calculates 2 raised to the power 3
  // In-built library function
  cout << pow(2, 3)<<endl;

  // Calling function factorise
  factorise(10);

  return 0;
}
```

(A) It calculates the power of 2 raised to the power of 3.

(B) It defines a user-defined function to find factors of a number and calls it to find factors of 10.

(C) It calculates the square root of 3.

(D) It defines a user-defined function to find the factorial of a number and calls it to find the factorial of 10.

Solution: (B) It defines a user-defined function to find factors of a number and calls it to find factors of 10.

Question 73:
Which of the following statements is true about the return type of a function in C++?

(A) The return type is optional; a function can choose not to have a return type.

(B) If the function does not return a value, its return type must be specified as void.

(C) Functions in C++ can have multiple return types.

(D) The return type is determined automatically by the compiler.

Solution: (B) If the function does not return a value, its return type must be specified as void.

Question 74:
What is the role of actual parameters in a function call?

(A) Actual parameters define the number and data types of input the function can have.

(B) Actual parameters are used in the function body to perform calculations.

(C) Actual parameters are used in the function declaration.

(D) Actual parameters are copied to formal parameters in the function definition.

Solution: (D) Actual parameters are copied to formal parameters in the function definition.

Question 75:
In C++, what happens if a function is called by value?

(A) Changes to the arguments are reflected outside the function.

(B) The arguments retain their original value outside the function.

(C) The function cannot modify the arguments.

(D) The arguments are passed by reference.

Solution: (B) The arguments retain their original value outside the function.

Question 76:
What is the primary benefit of user-defined functions in C++?

(A) They make the code longer and less readable.

(B) They cannot be reused in other programs.

(C) They allow for custom requirements and make the code more modular and readable.

(D) They increase the size of the code.

Solution: (C) They allow for custom requirements and make the code more modular and readable.

Question 77:
What is an inline function in C++?

(A) An inline function is a function that executes in a separate thread.

(B) An inline function is a function that is expanded in line when it is called, and its code is substituted at the point of the function call during compilation.

(C) An inline function is a function that contains a loop.

(D) An inline function is a function that returns a value other than void and does not have a return statement.

Solution: (B) An inline function is a function that is expanded in line when it is called, and its code is substituted at the point of the function call during compilation.

Question 78:
When is inlining a function beneficial in C++?

(A) For large functions that perform complex tasks.

(B) For small, commonly-used functions where the time needed to make the function call is significant compared to the execution time of the function's code.

(C) Inlining is always beneficial regardless of the size of the function.

(D) For functions containing loops, static variables, or recursive calls.

Solution: (B) For small, commonly-used functions where the time needed to make the function call is significant compared to the execution time of the function's code.

Question 79:
What is one advantage of using inline functions in C++?

(A) It increases function call overhead.

(B) It saves the overhead of push/pop variables on the stack during a function call.

(C) It limits the compiler's ability to perform context-specific optimization.

(D) It is useful for large functions with complex tasks.

Solution: (B) It saves the overhead of push/pop variables on the stack during a function call.

Question 80:
What is one disadvantage of using inline functions in C++?

(A) It reduces instruction cache hit rate.

(B) It increases function call overhead.

(C) It decreases the size of the binary executable file.

(D) It decreases compile time overhead.

Solution: (A) It reduces instruction cache hit rate.

Question 81:
When might inlining a function lead to a larger binary executable file?

(A) When the function is small and commonly used.

(B) When the function contains a loop.

(C) When too many inline functions are used.

(D) When the function is recursive.

Solution: (C) When too many inline functions are used.

Function overloading

Question 82:
What is function overloading in C++?

(A) It is a feature of object-oriented programming where two or more functions can have the same name and the same parameters.

(B) It is a feature of object-oriented programming where two or more functions can have the same name but different parameters.

(C) It is a feature of object-oriented programming where functions can have different names but the same parameters.

(D) It is a feature of object-oriented programming where functions can have different names and different parameters.

Solution: (B) It is a feature of object-oriented programming where two or more functions can have the same name but different parameters.

Question 83:
What is the primary benefit of function overloading in C++?

(A) It increases the complexity of the program.

(B) It reduces the readability of the program.

(C) It allows multiple functions with the same name but different parameters, improving code readability and maintainability.

(D) It allows multiple functions with different names and the same parameters.

Solution: (C) It allows multiple functions with the same name but different parameters, improving code readability and maintainability.

Question 84:
What are the conditions for function overloading?

(A) Functions should have the same name and the same parameters.

(B) Functions should have different names.

(C) Functions should have the same name but different parameters.

(D) Functions should have different names and different parameters.

Solution: (C) Functions should have the same name but different parameters.

Question 85:
Which of the following is a valid example of function overloading?
(A)

```
void add(int a, int b) { cout << "sum = " << (a + b); }
void add(int a, int b, int c) { cout << "sum = " << (a + b + c); }
```

(B)
```
void add(int a, int b) { cout << "sum = " << (a + b); }
void add(int c, int d) { cout << "sum = " << (c + d); }
```

(C)
```
void add(int a, int b) { cout << "sum = " << (a + b); }
void add(double a, double b) { cout << "sum = " << (a + b); }
```

(D)
```
void add(int a, int b) { cout << "sum = " << (a + b); }
void add(double a, int b) { cout << "sum = " << (a + b); }
```

Solution: (A)
```
void add(int a, int b) { cout << "sum = " << (a + b); }
void add(int a, int b, int c) { cout << "sum = " << (a + b + c); }
```

Question 86:
What is one disadvantage of function overloading?

(A) It improves code readability.

(B) It reduces the complexity of the program.

(C) It increases compile time overhead if changes are made to the inline function.

(D) It decreases the size of the binary executable file.

Solution: (C) It increases compile time overhead if changes are made to the inline function.

MCQs on C++ Pointers:

SWhat does the & operator provide in C++?

A. Address of a variable
B. Value of a variable
C. Pointer to a variable
D. Reference to a variable
Answer: A. Address of a variable
How do you declare a pointer variable in C++?

A. int varPointer;
B. pointer int var;
C. int* varPointer;
D. pointer var int;
Answer: C. int* varPointer;
Which operator is used for dereferencing a pointer in C++?

A. &
B. *
C. ->
D. ::
Answer: B. *
How do you assign the address of a variable var to a pointer ptr?

A. ptr = *var;
B. ptr = var;
C. ptr = &var;
D. ptr = address(var);
Answer: C. ptr = &var;

What is the output of the following code?

int num = 42;
int* pNum = &num;

cout << *pNum;

A. 42
B. &num
C. Memory address of num
D. Compiler Error
Answer: A. 42
In C++, what is the purpose of the new operator when working with pointers?

A. Create a new variable
B. Allocate memory for a variable
C. Delete a variable
D. Declare a new pointer
Answer: B. Allocate memory for a variable
What happens if you try to dereference a null pointer in C++?

A. Compile-time error
B. Runtime error
C. Undefined behavior
D. No effect
Answer: C. Undefined behavior
Which of the following conditions should be satisfied for function overloading in C++?

A. Same name and same parameters
B. Same name and different parameters
C. Different name and same parameters
D. Different name and different parameters
Answer: B. Same name and different parameters
What is the purpose of the delete operator when working with pointers in C++?

A. Free allocated memory
B. Delete a variable
C. Remove a pointer
D. Clear the pointer value
Answer: A. Free allocated memory
Which operator is used to find the address of a variable in C++?

A. *
B. &
C. ->
D. ::
Answer: B. &

MCQs on C++ OOPs Concepts:

What is the primary purpose of introducing object-oriented concepts to C++?

A. To make the language more complex
B. To enhance performance
C. To introduce features like inheritance and polymorphism
D. To support procedural programming
Answer: C. To introduce features like inheritance and polymorphism
Which programming paradigm represents everything as objects?

A. Procedural programming
B. Object-Oriented Programming
C. Functional programming
D. Logical programming
Answer: B. Object-Oriented Programming
What does OOP stand for?

A. Object-Oriented Procedure
B. Object-Oriented Programming
C. Only Object Programming
D. Object-Oriented Process
Answer: B. Object-Oriented Programming
Which of the following is an example of an object in the real world?

A. If statement
B. Integer variable
C. Pen
D. Loop
Answer: C. Pen
What is a collection of objects called in OOP?

A. Bundle
B. Array
C. Class
D. Ensemble
Answer: C. Class
In OOP, what does a class act as?

A. Object
B. Blueprint for objects
C. Data type
D. Function
Answer: B. Blueprint for objects

Which OOP concept provides the ability to simulate real-world events more effectively?

A. Inheritance
B. Polymorphism
C. Abstraction
D. Encapsulation
Answer: B. Polymorphism
What is the purpose of abstraction in OOP?

A. Showing internal details
B. Hiding internal details
C. Displaying all details
D. Encapsulating data
Answer: B. Hiding internal details
What does encapsulation in OOP involve?

A. Separating code and data
B. Binding code and data together
C. Exposing internal details
D. Enclosing data in a loop
Answer: B. Binding code and data together
What does dynamic binding in OOP refer to?

A. Code generation at runtime
B. Decision at runtime regarding code execution
C. Changing data type dynamically
D. Creating objects dynamically
Answer: B. Decision at runtime regarding code execution
Which advantage does OOP have over Procedure-oriented programming in terms of code management?

A. OOP makes it harder
B. OOP makes it easier
C. Both are the same
D. No impact on code management
Answer: B. OOP makes it easier
What does data hiding provide in OOP?

A. Global data access
B. Ability to hide global data
C. Global function access

D. Ability to hide local data
Answer: B. Ability to hide global data
Why does OOP provide a more effective solution to real-world problems?

A. Due to procedural programming
B. Due to global variables
C. Due to the ability to simulate real-world events
D. Due to syntax simplicity
Answer: C. Due to the ability to simulate real-world events
Why is C++ considered a partial OOP language?

A. It doesn't support encapsulation
B. The main function must be inside a class
C. It doesn't support polymorphism
D. It supports procedural programming only
Answer: B. The main function must be inside a class
What is the main function's role in making C++ a partial OOP language?

A. It breaks encapsulation
B. It allows for global variables
C. It requires a class for execution
D. It contradicts the concept of objects
Answer: C. It requires a class for execution
What is the term for an existing operator or function that is forced to operate on a new data type?

A. Polymorphism
B. Overloading
C. Encapsulation
D. Inheritance
Answer: B. Overloading
Which concept allows the reuse of code in OOP?

A. Encapsulation
B. Abstraction
C. Inheritance
D. Polymorphism
Answer: C. Inheritance
What is broken when global variables are used in C++?

A. Abstraction
B. Encapsulation

C. Inheritance
D. Polymorphism
Answer: B. Encapsulation
What is the process of tying together data and functions in object-oriented programming known as?

A. Abstraction
B. Encapsulation
C. Inheritance
D. Polymorphism
Answer: B. Encapsulation
Which feature allows the grouping of related information in OOP?

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

MCQs on Memory Management in C++:

What is the primary purpose of memory management in a computer system?

A. Speeding up computation
B. Managing storage devices
C. Assigning memory space to programs
D. Enhancing graphical user interfaces
Answer: C. Assigning memory space to programs
Why is dynamic memory allocation required in programming?

A. To speed up compilation
B. To allocate memory at declaration time
C. To avoid wastage of memory
D. To improve graphical performance
Answer: C. To avoid wastage of memory
Which operators are used for dynamic memory allocation in C++?

A. alloc and dealloc
B. malloc and free
C. new and delete

D. allocate and deallocate
Answer: C. new and delete
What is the purpose of the new operator in C++?

A. To create a new variable
B. To allocate dynamic memory
C. To perform arithmetic operations
D. To define a new function
Answer: B. To allocate dynamic memory
What is the syntax for dynamically creating an integer object using the new operator?

A. int *p = new int;
B. new int *p;
C. p = new int;
D. p = new int();
Answer: A. int *p = new int;
How is the value assigned to a dynamically created object using the new operator?

A. *p = 5;
B. p = 5;
C. value(p) = 5;
D. p = new int(5);
Answer: A. *p = 5;
What is the purpose of the delete operator in C++?

A. To remove a variable
B. To deallocate static memory
C. To free up dynamic memory
D. To delete a function
Answer: C. To free up dynamic memory
How do you delete a dynamically allocated integer object pointed by p?

A. remove p;
B. delete p;
C. free p;
D. p = nullptr;
Answer: B. delete p;
What is the correct syntax to delete a dynamically allocated array arr?

A. delete arr;
B. free arr;
C. delete [] arr;

D. arr = nullptr;
Answer: C. delete [] arr;
What advantage does the new operator have over malloc() function in C++?

A. Uses sizeof() automatically
B. Requires typecasting
C. Only allocates memory for primitive types
D. Doesn't support dynamic memory allocation
Answer: A. Uses sizeof() automatically

MCQs on Pointers and Dynamic Memory Management in C++:

What is the purpose of the new operator in C++?

A. To declare a new variable
B. To allocate memory dynamically
C. To create a new function
D. To initialize a pointer
Answer: B. To allocate memory dynamically

How is memory deallocated in C++ after using new for memory allocation?

A. free keyword
B. dispose function
C. delete operator
D. release method
Answer: C. delete operator

What is a common issue associated with dangling pointers?

A. Memory leaks
B. Uninitialized pointers
C. Pointers pointing to invalid memory
D. Null pointer assignment
Answer: C. Pointers pointing to invalid memory

What does the this pointer refer to in C++?

A. The current date and time
B. The class instance for which a member function is called
C. The address of the first element in an array

D. A pointer to the previous object
Answer: B. The class instance for which a member function is called

How do you avoid memory leaks in C++?

A. Using uninitialized pointers
B. Handling null pointer assignments
C. Properly deallocating memory using delete or delete[]
D. Creating wild pointers
Answer: C. Properly deallocating memory using delete or delete[]

What is a wild pointer in C++?

A. A pointer that points to an invalid memory location
B. A pointer with a null value
C. A pointer that is not initialized
D. A pointer that points to a valid object
Answer: C. A pointer that is not initialized

When is a std::bad_alloc exception thrown in C++?

A. When a null pointer is assigned
B. When there is insufficient memory for a new request
C. When using the delete operator
D. When handling exceptions in a function
Answer: B. When there is insufficient memory for a new request

What does a null pointer assignment in C++ mean?

A. Assigning a pointer to a valid memory location
B. Initializing a pointer with a random value
C. Assigning a null value (0 or nullptr) to a pointer
D. Assigning a pointer to another pointer
Answer: C. Assigning a null value (0 or nullptr) to a pointer

How can smart pointers help in C++ memory management?

A. They prevent null pointer assignments
B. They automatically handle memory deallocation
C. They avoid wild pointer issues
D. All of the above
Answer: D. All of the above

What is the purpose of the delete[] operator in C++?

A. To delete an individual object
B. To delete a pointer
C. To delete an array of objects
D. To delete a class definition
Answer: C. To delete an array of objects

MCQs on Function Call Overhead and Inline Functions in C++:

What is function call overhead in C++?

A. The time taken to execute a function
B. The performance cost associated with calling a function
C. The memory consumed by a function
D. The return value of a function
Answer: B. The performance cost associated with calling a function

Which of the following is NOT a part of function call overhead?

A. Pushing parameters
B. Setting up the stack frame
C. Jumping to the function
D. Executing the function code
Answer: D. Executing the function code

When is function call overhead likely to become significant?

A. When functions are called in tight loops
B. When functions have large stack frames
C. When functions are small
D. When functions use recursion
Answer: A. When functions are called in tight loops

What is the purpose of the inline keyword in C++?

A. To define a new function
B. To suggest the compiler to perform inline expansion of a function
C. To create a function pointer
D. To declare a recursive function

Answer: B. To suggest the compiler to perform inline expansion of a function

How does an inline function differ from a regular function in terms of performance?

A. Inline functions have higher performance always
B. Regular functions have higher performance always
C. Both have the same performance
D. It depends on the size and complexity of the function
Answer: A. Inline functions have higher performance always

In C++, when is the compiler likely to ignore the inline keyword?

A. When the function is small
B. When the function is large
C. When the function is declared in a header file
D. When the function contains a return statement
Answer: B. When the function is large

What is one of the advantages of using inline functions in C++?

A. Increased memory usage
B. Code bloat
C. Reduced function call overhead
D. Limited optimization
Answer: C. Reduced function call overhead

Where are inline functions often defined in C++?

A. Source files
B. Standalone header files
C. Shared libraries
D. Binary executables
Answer: B. Standalone header files

What happens when an inline function is included in multiple source files?

A. It leads to a compilation error
B. Each source file gets its own copy of the inline function
C. The function is shared across source files
D. It results in a runtime error
Answer: B. Each source file gets its own copy of the inline function

What is a key consideration when using the inline keyword in C++?

A. Maximizing code bloat
B. Optimizing all functions
C. Using inline for large functions
D. Using inline judiciously
Answer: D. Using inline judiciously

MCQs on Destructors in C++:

What is a destructor in C++?

A. A member function that initializes the object
B. A special member function that is invoked before the object is created
C. A member function that deallocates the memory occupied by the object
D. A function that creates objects of a class
Answer: C. A member function that deallocates the memory occupied by the object

Which symbol precedes the name of a destructor in C++?

A. @
B. #
C. ~
D. $
Answer: C. ~

Can a class have more than one destructor?

A. Yes, but only in specific cases
B. No, a class can have only one destructor
C. Yes, always
D. Yes, if they have different parameters
Answer: B. No, a class can have only one destructor

What is the role of a destructor in C++?

A. Initializing objects
B. Deallocating memory and cleaning up resources
C. Creating objects
D. Assigning values to class members

Answer: B. Deallocating memory and cleaning up resources

When is a destructor called in C++?

A. Before an object is created
B. After an object is created
C. When the program starts
D. When an object is going to be destroyed
Answer: D. When an object is going to be destroyed

Which section of the class should the destructor be declared in?

A. Public
B. Private
C. Protected
D. Any of the above
Answer: A. Public

Can a destructor have arguments in C++?

A. Yes
B. No
C. Only if it has a return type
D. Only if it is static
Answer: B. No

How is memory released in a user-defined destructor when dealing with dynamic memory allocation?

A. Using delete operator
B. Using free() function
C. Automatically by the compiler
D. Manually by the programmer
Answer: A. Using delete operator

When should you write a user-defined destructor in C++?

A. Always
B. Only if there are no pointers in the class
C. Only if the class has no member functions
D. When the class contains dynamically allocated memory or pointers
Answer: D. When the class contains dynamically allocated memory or pointers

How can a destructor be called explicitly in C++?

A. Using the delete keyword
B. By invoking it like a regular member function
C. Automatically when an object goes out of scope
D. By passing arguments to it
Answer: B. By invoking it like a regular member function

MCQs on C++ Containers:

Which of the following is a sequence container in C++?

A. Set
B. Vector
C. Map
D. Multiset
Answer: B. Vector

What is the purpose of an associative container in C++?

A. Storing data in a linear fashion
B. Fast access to elements
C. Modeling real-life scenarios
D. Deriving other containers
Answer: B. Fast access to elements

Which of the following is an example of an associative container in C++?

A. Stack
B. Queue
C. List
D. Map
Answer: D. Map

Derived containers in C++ are:

A. Used for linear data storage
B. Always based on sequence containers
C. Derived from either sequence or associative containers
D. Limited to stack and queue
Answer: C. Derived from either sequence or associative containers

In an associative container, how is data typically stored?

A. Linear fashion
B. Tree-like structure
C. In a stack
D. As a queue
Answer: B. Tree-like structure

Which container is specifically designed for fast access to elements in C++?

A. Vector
B. List
C. Dequeue
D. Set
Answer: D. Set

What kind of data structure is often used by associative containers in C++?

A. Linked list
B. Tree
C. Array
D. Queue
Answer: B. Tree

Which of the following is a sequence container used in C++?

A. Stack
B. Map
C. List
D. Priority Queue
Answer: C. List

Derived containers in C++ provide:

A. Fast access to elements
B. Better methods for dealing with data
C. Linear storage of data
D. No additional methods
Answer: B. Better methods for dealing with data

Which container is used for modeling real-life scenarios and follows the First In First Out (FIFO) principle?

A. Set
B. Queue
C. Multiset
D. Dequeue
Answer: B. Queue

MCQs on Vectors in C++:

What is a vector in C++?

A. A graphical representation
B. A linear sequence container
C. A mathematical function
D. A character data type
Answer: B. A linear sequence container

Which header file must be included to use vectors in C++?

A. <array>
B. <list>
C. <vector>
D. <iostream>
Answer: C. <vector>

How are elements accessed in a vector compared to arrays?

A. Slower
B. Faster
C. Similar
D. Not possible in vectors
Answer: B. Faster

What is the primary advantage of using vectors in C++?

A. Faster insertion at random positions
B. Faster deletion at random positions
C. Faster access to elements

D. Smaller memory footprint
Answer: C. Faster access to elements

Which of the following is a correct way to initialize a vector with elements 1, 2, 3, and 4?

A. vector<int> v = {1, 2, 3, 4};
B. vector<int> v = (1, 2, 3, 4);
C. vector<int> v(1, 2, 3, 4);
D. vector<int> v(4, 1);
Answer: A. vector<int> v = {1, 2, 3, 4};

What data types can be stored in a vector in C++?

A. Only integers
B. Only characters
C. Elements of a single data type
D. Elements of multiple data types
Answer: C. Elements of a single data type

How can a vector be initialized with a specific size and a uniform value?

A. vector<int> v(size, value);
B. vector<int> v(size = value);
C. vector<int>(value, size);
D. vector<int>(size, value);
Answer: D. vector<int>(size, value);

MCQs on Lists in C++:

What is a primary characteristic of lists in C++ compared to vectors?

A. Contiguous memory allocation
B. Slow insertion and deletion
C. Fast traversal
D. Linear sequence container
Answer: B. Slow insertion and deletion

Which header file should be included to use lists in C++?

A. <array>
B. <list>
C. <vector>
D. <iostream>
Answer: B. <list>

What is the default type of a list in C++?

A. array
B. vector
C. list
D. linked list
Answer: C. list

Which of the following operations is slow in a list compared to a vector?

A. Traversal
B. Insertion at random position
C. Deletion at random position
D. Accessing elements
Answer: A. Traversal

What is the primary advantage of using a list over a vector in C++?

A. Faster traversal
B. Faster insertion at random positions
C. Faster deletion at random positions
D. Smaller memory footprint
Answer: C. Faster insertion at random positions

Which function is used to add a new element at the beginning of a list?

A. push_back()
B. pop_front()
C. push_front()
D. insert()
Answer: C. push_front()

What does pop_back() do in a list?

A. Adds an element at the end
B. Removes the first element

C. Removes the last element
D. Adds an element at the beginning
Answer: C. Removes the last element

Which function is used to insert new elements before the element at a specified position in a list?

A. insert()
B. push_back()
C. push_front()
D. pop_back()
Answer: A. insert()

What does the size() function return for a list?

A. Total available memory
B. Number of elements in the list
C. Maximum size of the list
D. Capacity of the list
Answer: B. Number of elements in the list

What does the end() function return for a list in C++?

A. Iterator pointing to the last element
B. Iterator pointing to the first element
C. Iterator pointing to the theoretical last element
D. Iterator pointing to a random element
Answer: C. Iterator pointing to the theoretical last element