Engineering License Exam Preparation: Programming Language and Its Applications (ACtE03)

Section 3.1: Introduction to C Programming

Overview

The C programming language is one of the foundational programming languages, particularly in system programming and low-level applications. This section introduces key concepts such as tokens, operators, control structures, and essential data types like arrays and strings. Understanding these concepts is crucial for building algorithms and solving problems using structured programming.

Key Concepts

- **C Tokens**: Keywords, Identifiers, Constants, Strings, Operators.
- **Operators**: Arithmetic, Relational, Logical, Bitwise, Assignment, Increment/Decrement.
- **Input/Output**: Formatted (`printf`, `scanf`) and Unformatted (`getchar`, `putchar`).
- **Control Statements**: `if`, `else`, `switch`.
- **Looping Statements**: `for`, `while`, `do-while`.
- **Functions**: User-defined and recursive functions.
- **Arrays**: One-dimensional (1-D), Two-dimensional (2-D), Multi-dimensional arrays.
- **String Manipulation**: Functions like `strlen()`, `strcpy()`, `strcmp()`, and others.

Most Probable MCQs for Section 3.1

1. **Q1**: What is a C token?
   A) Keyword
   B) Constant
   C) Identifier
   D) All of the above
   Answer: D) All of the above

2. **Q2**: What is the output of the following code?

   ```
   int a = 5, b = 10;
   printf("%d", a > b ? a : b);
   ```

   A) 5
   B) 10
   C) Error
   D) None
   Answer: B) 10

3. **Q3**: Which control statement is used to exit from a loop?
   A) `break`
   B) `continue` C) `goto`
   D) `return`
   Answer: A) `break`

4. **Q4**: Which data type is used to store a string in C?
   A) `char[]`
   B) `int[]`
   C) `float[]`
   D) `double[]`
   Answer: A) `char[]`

5. **Q5**: Which operator is used to access the elements of an array?
   A) `.`
   B) `->`
   C) `[]`
   D) `*`
   Answer: C) `[]`

Section 3.2: Pointers, Structure, and Data Files in C Programming

Overview

Pointers and structures are essential for handling dynamic memory and organizing complex data structures in C programming. This section also covers file handling, which is critical for reading and writing data in persistent storage.

Key Concepts

- **Pointers**: Memory addresses, pointer arithmetic, passing pointers to functions.
- **Structures vs Unions**: Structures allow grouping different data types, while unions share memory between members.
- **Pointer and Array**: Relationship between pointers and arrays.
- **File Handling**: Input/output operations, sequential and random access to files, `fopen`, `fread`, `fwrite`.

Most Probable MCQs for Section 3.2

6. **Q6**: What does a pointer store in C?
   A) Value of a variable
   B) Address of a variable
   C) Array of data
   D) None of the above
   Answer: B) Address of a variable

7. **Q7**: Which function is used to open a file in C?
   A) `fwrite()`
   B) `fopen()`

C) `fclose()`
D) `fseek()`
Answer: B) `fopen()`

8. **Q8**: Which is the correct way to declare a pointer?
   A) `int *ptr;`
   B) `int ptr*;`
   C) `*int ptr;`
   D) `ptr int*;`
   Answer: A) `int *ptr;`

9. **Q9**: Which is correct about structure vs union?
   A) Structure uses shared memory
   B) Union uses shared memory
   C) Both use shared memory
   D) Neither uses shared memory
   Answer: B) Union uses shared memory

10. **Q10**: Which of the following operations is not allowed on pointers?
    A) Addition
    B) Subtraction
    C) Multiplication
    D) Comparison
    Answer: C) Multiplication

Section 3.3: C++ Language Constructs with Objects and Classes

Overview

C++ is an extension of C that introduces object-oriented programming (OOP). This section covers the basics of OOP, including classes, objects, and essential OOP features such as function overloading and dynamic memory allocation.

Key Concepts

- **Namespace**: Defines a scope for identifiers to avoid name conflicts.
- **Function Overloading**: Multiple functions with the same name but different parameters.
- **Inline Functions**: Functions that are expanded in line where they are invoked.
- **Classes and Objects**: Basic constructs of OOP.
- **Constructors and Destructors**: Special functions for object initialization and cleanup.

Most Probable MCQs for Section 3.3

11. **Q11**: What is the default access specifier for members of a class in C++?
    A) `public`
    B) `private`
    C) `protected`

D) `friend`
Answer: B) `private`

12. **Q12**: Which function is called when an object is destroyed in C++?
    A) Constructor
    B) Destructor
    C) Friend function
    D) Inline function
    Answer: B) Destructor

13. **Q13**: What does the `this` pointer point to?
    A) The class name
    B) The calling object
    C) The class constructor
    D) A local variable
    Answer: B) The calling object

14. **Q14**: How do you define a dynamic array of objects in C++?
    A) `new Object[n];`
    B) `Object[] arr;`
    C) `Object *arr = new Object[n];`
    D) `Object arr = new Object[];`
    Answer: C) `Object *arr = new Object[n];`

15. **Q15**: Which of the following allows overloading of functions?
    A) Default arguments
    B) Different return types
    C) Different parameter lists
    D) Both A and C
    Answer: D) Both A and C

Section 3.4: Features of Object-Oriented Programming

Overview

OOP is a paradigm that organizes software design around data (objects) and operations (methods). This section covers advanced OOP concepts, including operator overloading, inheritance, and constructors/destructors in inheritance.

Key Concepts

- **Operator Overloading**: Defining custom behavior for operators.
- **Inheritance**: Deriving new classes from existing ones.
  - **Types of Inheritance**: Single, multiple, multilevel, hybrid, multipath.
- **Constructors/Destructors in Inheritance**: Special rules for constructor and destructor calls in derived classes.

Most Probable MCQs for Section 3.4

16. **Q16**: Which type of inheritance involves one base class and multiple derived classes?
    A) Single
    B) Multiple
    C) Multilevel
    D) Hybrid
    Answer: A) Single

17. **Q17**: Which operator can be overloaded in C++?
    A) `sizeof`
    B) `&&`
    C) `*`
    D) `?:`
    Answer: C) `*`

18. **Q18**: What is the order of constructor calls in multilevel inheritance?
    A) Base class first
    B) Derived class first
    C) Virtual base class first
    D) Arbitrary order
    Answer: A) Base class first

19. **Q19**: In which type of inheritance is a class derived from two or more base classes?
    A) Single inheritance
    B) Multiple inheritance
    C) Multilevel inheritance
    D) Hybrid inheritance
    Answer: B) Multiple inheritance

20. **Q20**: What is the purpose of a destructor in inheritance?
    A) To allocate resources
    B) To free resources
    C) To overload functions
    D) To define classes
    Answer: B) To free resources

Section 3.5: Pure Virtual Functions and File Handling

Overview

Virtual functions enable dynamic (runtime) polymorphism. This section also deals with file handling operations, which are vital for input/output (I/O) in persistent storage.

Key Concepts

- **Virtual Functions**: Functions that can be overridden in derived classes.
- **File Handling**: Reading and writing to files, error handling in I/O operations

.

Most Probable MCQs for Section 3.5

21. **Q21**: What is a pure virtual function in C++?
    A) A function with no body
    B) A function with arguments
    C) A function with a return type
    D) A function with no return type
    Answer: A) A function with no body

22. **Q22**: Which class is used for file handling in C++?
    A) `fstream`
    B) `istream`
    C) `iostream`
    D) `string`
    Answer: A) `fstream`

23. **Q23**: Which function is used to write data to a file?
    A) `fwrite()`
    B) `write()`
    C) `fput()`
    D) `putdata()`
    Answer: A) `fwrite()`

24. **Q24**: What is the default mode for opening a file in C++?
    A) Input
    B) Output
    C) Append
    D) Binary
    Answer: A) Input

25. **Q25**: How is an error handled during input/output operations in C++?
    A) Using exceptions
    B) Using error codes
    C) Using `try-catch` blocks
    D) Both A and C
    Answer: D) Both A and C

Section 3.6: Generic Programming and Exception Handling

Overview

Generic programming allows code reusability using templates, and exception handling ensures that runtime errors are managed gracefully without crashing the program.

Key Concepts

- **Templates**: Class and function templates.
- **Exception Handling**: `try`, `catch`, `throw` mechanisms.

- **Standard Template Library (STL)**: Containers, algorithms, and iterators.

Most Probable MCQs for Section 3.6

26. **Q26**: Which of the following is true about function templates?
    A) They are used to create functions for specific types
    B) They allow functions to work with any data type
    C) They cannot be overloaded
    D) They do not support default arguments
    Answer: B) They allow functions to work with any data type

27. **Q27**: What is the keyword used to catch exceptions in C++?
    A) `try`
    B) `throw`
    C) `catch`
    D) `finally`
    Answer: C) `catch`

28. **Q28**: Which container in the STL uses a last-in, first-out (LIFO) structure?
    A) Vector
    B) Queue
    C) Stack
    D) List
    Answer: C) Stack

29. **Q29**: What happens if an exception is not caught in C++?
    A) Program continues
    B) Program terminates
    C) Program shows a warning
    D) None of the above
    Answer: B) Program terminates

30. **Q30**: Which keyword is used to rethrow an exception in C++?
    A) `rethrow`
    B) `retry`
    C) `throw`
    D) `exception`
    Answer: C) `throw`
