C-DAC MUMBAI

Object Oriented Programming using C++

Assigment-2

Q1.1 Create a class `Box` with private members `length`, `width`, and `height`.

Write:

- A parameterized constructor
- A function `setDimensions(int, int, int)` that uses `this->` to distinguish between member variables and parameters
- A function `volume()` to compute the volume

Demonstrate:

- Initialization using constructor
- Assignment using `setDimensions()`

Q1.2 — Answer this:

- Why must initialization happen before assignment?
- When is initialization preferred over assignment?

Q2.1 Write three functions:

```
void swapByValue(int a, int b);
void swapByAddress(int *a, int *b);
void swapByReference(int &a, int &b);
```

Call all three in `main()` and observe which one actually swaps values.

Q2.2 — Answer this:

Explain how reference variables act as aliases and how that affects `swapByReference()`.

Q3.1 Write a program to store an integer value in a variable, then:

- Create a pointer pointing to the variable
- Create a reference to the same variable
- Modify the value using pointer and reference

Print the variable after each change.

Q3.2 — Explain:

- Two differences between pointer and reference
- Why references cannot be reseated but pointers can
- Why references cannot be NULL

Q4.1 Write a program that:

- Uses `new` to allocate an array of 5 integers
- Takes user input
- Prints the values
- Deallocates memory using `delete[]`

Q4.2 Repeat Q4.1 using `malloc` and `free`.

Q4.3 — Explain:

- Why constructors are not called when using `malloc`
- Why `new` is preferred in C++
- Difference in return types and initialization
- Why `malloc` cannot initialize complex types

Q5.1 Create a class `Student` with:

- rollNo
- name
- marks

Write the following:

1. Default constructor
2. Parameterized constructor
3. Constructor that uses `this->` pointer
4. A function to print student details

Create:

- One object using default constructor
- Two objects using parameterized constructor

Q5.2 — Answer:

- When does compiler generate a default constructor?
- When does it NOT generate one?
- Can constructors be overloaded?

Q6.1 Create a class `Employee` with:

- const int employeeId
- string name
- float salary

Write a constructor using **initializer list** to initialize all members.

Q6.2 — Add a function to display details.

Q6.3 — Answer these:

1. Why must `const` members be initialized in initializer list?
2. What happens if you try to assign the value of a const member inside constructor body?
3. Why is initializer list faster than assignment?