C and C++ Programming Assessment 2 Answers

June 12, 2025

C Multiple Choice Answers

1. **B) 4 bytes**

   A `float` typically occupies 4 bytes on most systems, following the IEEE 754 single-precision format.

2. **B) variable$_1$**

   C variable names must begin with a letter or underscore and can include letters, digits, and underscores. Option A starts with a digit, C uses a hyphen, and D uses a hash, all invalid.

3. **A) 10**

   The post-increment ‘x++‘ evaluates to 5 (current value of ‘x‘), then increments ‘x‘ to 6. Thus, ‘5 * 2 = 10‘ is printed.

4. **B) Buffer overflow**

   The string "Hello World" (11 characters + null terminator = 12 bytes) exceeds the ‘str[10]‘ capacity, causing a buffer overflow. Assumes ‘¡string.h¿‘ is included.

5. **C) for**

   The ‘for‘ loop is ideal for arrays of known size, allowing clear index initialization, condition checking, and incrementation.

6. **C) Direct assignment of another array**

   Arrays in C cannot be assigned directly (e.g., ‘arr1 = arr2‘). Initialization requires static, dynamic, or element-by-element methods.

7. **B) Out-of-bounds access**

   The array ‘arr[3]‘ has indices 0 to 2. Accessing ‘*(ptr + 3)‘ (index 3) is out of bounds, causing undefined behavior.

8. **B) Preserves variable value between function calls**

   A ‘static‘ variable inside a function retains its value across calls, unlike automatic variables.

9. **B) int (*func)();**

   A function pointer is declared as ‘int (*func)()‘, where ‘*func‘ points to a function returning ‘int‘ with no parameters.

10. **B) Writing to a closed file**

    The file is closed with ‘fclose(fp)‘ before ‘fprintf‘, causing undefined behavior when writing to a closed file pointer.

C++ Multiple Choice Answers

1. **D) internal**

   C++ access specifiers are ‘public‘, ‘private‘, and ‘protected‘. ‘internal‘ is not a C++ keyword.

2. **A) Missing semicolon after class**

   A class definition requires a semicolon after the closing brace. The code also lacks an access specifier, but the primary syntax error is the missing semicolon.

3. **B) Dynamic binding**

   The ‘virtual‘ keyword enables dynamic binding, allowing runtime polymorphism via virtual functions.

4. **B) Constructors cannot return values**

   Constructors initialize objects and cannot have a return type or return values, even implicitly.

5. **B) Deallocates memory on the heap**

   The ‘delete‘ operator frees memory allocated on the heap using ‘new‘.

6. **A) Out-of-bounds access**

   The array ‘arr[5]‘ has indices 0 to 4. Accessing ‘arr[5]‘ is out of bounds, causing undefined behavior.

7. **B) Creates a copy of an existing object**

   A copy constructor initializes a new object as a copy of an existing object of the same class.

8. **B) const**

   A ‘const‘ member function, declared with ‘const‘ after the function signature, cannot modify the object's state.

9. **A) Single, Multiple, Multilevel, Hierarchical, Hybrid**

   These are the types of inheritance in C++. Option B refers to access specifiers, not inheritance types.

10. **B) Class B**

    Due to the ‘virtual‘ function and dynamic binding, the ‘display‘ function of the derived class ‘B‘ is called via the base class pointer, outputting "Class B".

C Creative Question Answers

1. **Array Definition and Printing**

```
        int main() {
            int arr[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
            for (int i = 0; i < 10; i++) {
                printf("%d ", arr[i]);
            }
            printf("\n");
            return 0;
        }
```

   Initializes the array and prints all elements to confirm.

2. **Binary Search Function**

```
        int binarySearch(int arr[], int size, int value) {
            int left = 0, right = size - 1;
            while (left <= right) {
                int mid = left + (right - left) / 2;
                if (arr[mid] == value) return mid;
                if (arr[mid] < value) left = mid + 1;
                else right = mid - 1;
            }
            return -1;
        }
```

   Implements binary search on a sorted array, returning the index or -1 if not found.

3. **Modify main with Binary Search**

```c
int main() {
    int arr[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    for (int i = 0; i < 10; i++) {
        printf("%d ", arr[i]);
    }
    printf("\nEnter value to search: ");
    int value;
    scanf("%d", &value);
    int result = binarySearch(arr, 10, value);
    if (result != -1)
        printf("Value %d found at index %d\n", value, result);
    else
        printf("Value %d not found\n", value);
    return 0;
}

```

Prompts user input, calls ‘binarySearch‘, and prints the result.

4. **Find Second Largest**

```c
int findSecondLargest(int arr[], int size) {
    int first = arr[0], second = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > first) {
            second = first;
            first = arr[i];
        } else if (arr[i] > second && arr[i] != first) {
            second = arr[i];
        }
    }
    return second;
}

```

Finds the second largest value, called in ‘main‘ to print the result.

5. **Calculate Median**

```c
float calculateMedian(int arr[], int size) {
    return (float)(arr[size/2 - 1] + arr[size/2]) / 2;
}

```

For an even-sized sorted array, returns the average of the two middle elements.

C++ Creative Question Answers

1. **Employee Class Definition**

```cpp
class Employee {
private:
    string name;
    float salaries[6];
    int id;
public:
    Employee(string n, float s[], int i) {
        name = n;
        for (int j = 0; j < 6; j++) salaries[j] = s[j];
        id = i;
    }
};

```

Defines the class with private members and a constructor.

2. **Calculate Average Salary**

```
        float calculateAverageSalary() {
            float sum = 0;
            for (int i = 0; i < 6; i++) {
                sum += salaries[i];
            }
            return sum / 6;
        }

```

Computes and returns the average of the salaries.

3. **Is Above Threshold**

```
        bool isAboveThreshold() {
            return calculateAverageSalary() > 5000;
        }

```

Returns ‘true‘ if the average salary exceeds 5000.

4. **Display Info**

```
        void displayInfo() {
            cout << "Name: " << name << endl;
            cout << "ID: " << id << endl;
            cout << "Salaries: ";
            for (int i = 0; i < 6; i++) {
                cout << salaries[i] << " ";
            }
            cout << endl;
            float avg = calculateAverageSalary();
            cout << "Average Salary: " << fixed << setprecision(2) << avg <<
    endl;
            cout << "Status: " << (isAboveThreshold() ? "Above Threshold" : "
    Below Threshold") << endl;
        }

```

Displays all employee details in the specified format.

5. **Main Function**

```
        int main() {
            float salaries1[6] = {4500.0, 4700.0, 4800.0, 4900.0, 5000.0,
    5100.0};
            float salaries2[6] = {5500.0, 5600.0, 5700.0, 5800.0, 5900.0,
    6000.0};
            Employee emp1("Alice Johnson", salaries1, 201);
            Employee emp2("Bob Wilson", salaries2, 202);
            emp1.displayInfo();
            cout << endl;
            emp2.displayInfo();
            return 0;
        }

```

Creates two ‘Employee‘ objects and calls ‘displayInfo‘ to test all functionalities.