C and C++ Programming Assessment 3 Answers

June 12, 2025

C Multiple Choice Answers

1. **B) 8 bytes**

   A `double` typically occupies 8 bytes on most systems, following the IEEE 754 double-precision format.

2. **C) string**

   C does not have a built-in `string` data type; strings are handled as arrays of `char`. Other options are valid C types.

3. **D) Undefined Behavior**

   The expression `++x + x++` modifies `x` multiple times without a sequence point, leading to undefined behavior.

4. **A) Pointer not initialized**

   The pointer `ptr` is not initialized before dereferencing, causing undefined behavior when assigning `*ptr = 10`.

5. **D) Both B and C**

   Both `while` and `for` loops check their condition before executing the body, unlike `do-while`.

6. **C) Size of a pointer**

   When an array is passed to a function, it decays to a pointer, so `sizeof` returns the size of the pointer (typically 4 or 8 bytes).

7. **A) Loop condition is incorrect**

   The loop condition `i <= 5` accesses `arr[5]`, which is out of bounds for `arr[5]` (indices 0 to 4).

8. **A) Defines a variable as immutable**

   The `const` keyword declares a variable as read-only, preventing modification after initialization.

9. **B) int arr[3][3];**

   A 2D array is declared as `int arr[rows][cols]`. Option A uses incorrect syntax, C is invalid, and D declares an array of pointers.

10. **A) Missing variable in fscanf**

    `fscanf` requires a variable to store the scanned value (e.g., `fscanf(fp, "%d", &x)`), otherwise it causes undefined behavior.

C++ Multiple Choice Answers

1. **B) References**

   References are a C++ feature, not available in C. Pointers, arrays, and structures exist in both languages.

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

   A class definition requires a semicolon after the closing brace. The syntax for `public` is correct but lacks the semicolon.

3. **B) Function overrides a base class virtual function**

   The `override` keyword ensures a function overrides a virtual function in the base class, preventing errors.

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

   Constructors initialize objects and cannot return values, even local variables like `x`.

5. **B) Current object instance**

   The `this` pointer refers to the current object instance within a class's member functions.

6. **A) Incorrect deletion syntax for single object**

   A single object allocated with `new` should be deleted with `delete`, not `delete[]`, which is for arrays.

7. **A) To redefine operators for user-defined types**

   Operator overloading allows custom behavior for operators (e.g., `+`) with user-defined classes.

8. **B) final**

   The `final` keyword prevents a class from being inherited or a virtual function from being overridden.

9. **A) Default access specifier**

   In a `struct`, members are `public` by default; in a `class`, they are `private`.

10. **B) Derived**

    The `virtual` function enables dynamic binding, so the `print` function of `Derived` is called, outputting "Derived".

C Creative Question Answers

1. **Array Definition and Printing**

```
        int main() {
            int arr[8] = {5, 10, 15, 20, 25, 30, 35, 40};
            for (int i = 0; i < 8; i++) {
                printf("%d ", arr[i]);
            }
            printf("\n");
            return 0;
        }

```

   Initializes and prints the array to confirm.

2. **Reverse Array Function**

```
        void reverseArray(int arr[], int size) {
            for (int i = 0; i < size / 2; i++) {
                int temp = arr[i];
                arr[i] = arr[size - 1 - i];
                arr[size - 1 - i] = temp;
            }
        }

```

   Reverses the array in place by swapping elements from the ends.

3. **Modify main with Reverse Array**

```
        int main() {
            int arr[8] = {5, 10, 15, 20, 25, 30, 35, 40};
            printf("Original array: ");
            for (int i = 0; i < 8; i++) {
                printf("%d ", arr[i]);
            }
```

```
    printf("\n");
    reverseArray(arr, 8);
    printf("Reversed array: ");
    for (int i = 0; i < 8; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Calls `reverseArray` and prints the reversed array.

4. **Find Duplicates**

```
void findDuplicates(int arr[], int size) {
    int found = 0;
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                printf("Duplicate found: %d\n", arr[i]);
                found = 1;
            }
        }
    }
    if (!found) printf("No duplicates found\n");
}
```

Checks for duplicates and prints them; called in `main`.

5. **Calculate Range**

```
int calculateRange(int arr[], int size) {
    int max = arr[0], min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) max = arr[i];
        if (arr[i] < min) min = arr[i];
    }
    return max - min;
}
```

Returns the range (max - min), printed in `main`.

C++ Creative Question Answers

1. **Book Class Definition**

```
class Book {
private:
    string title;
    float ratings[4];
    int isbn;
public:
    Book(string t, float r[], int i) {
        title = t;
        for (int j = 0; j < 4; j++) ratings[j] = r[j];
        isbn = i;
    }
};
```

Defines the class with private members and a constructor.

2. **Calculate Average Rating**

```
        float calculateAverageRating() {
            float sum = 0;
            for (int i = 0; i < 4; i++) {
                sum += ratings[i];
            }
            return sum / 4;
        }

```

Computes and returns the average rating.

3. **Is Highly Rated**

```
        bool isHighlyRated() {
            return calculateAverageRating() >= 4.0;
        }

```

Returns `true` if the average rating is 4.0 or higher.

4. **Display Book Info**

```
        void displayBookInfo() {
            cout << "Title: " << title << endl;
            cout << "ISBN: " << isbn << endl;
            cout << "Ratings: ";
            for (int i = 0; i < 4; i++) {
                cout << ratings[i] << " ";
            }
            cout << endl;
            float avg = calculateAverageRating();
            cout << "Average Rating: " << fixed << setprecision(2) << avg <<
    endl;
            cout << "Status: " << (isHighlyRated() ? "Highly Rated" : "Not
    Highly Rated") << endl;
        }

```

Displays book details in the specified format.

5. **Main Function**

```
        int main() {
            float ratings1[4] = {4.5, 4.0, 3.8, 4.2};
            float ratings2[4] = {3.5, 3.0, 3.2, 3.8};
            Book book1("The Great Gatsby", ratings1, 123456789);
            Book book2("1984", ratings2, 987654321);
            book1.displayBookInfo();
            cout << endl;
            book2.displayBookInfo();
            return 0;
        }

```

Creates two `Book` objects and tests all functionalities.