C++ Concepts Review Guide

Quiz

Answer the following questions in 2-3 sentences each.

1. Explain the purpose of the relational() function.
2. Describe how a simple ascending star pattern is generated using nested loops, as seen in multiple examples.
3. What is function overloading, and where is an example of it in the provided code related to the sereveChai function?
4. Explain the concept of a "copy constructor" in C++ with reference to the Chai class example.
5. How is dynamic memory allocation demonstrated in the prepareOrder function (the version using int* orders)? What is its importance?
6. What is the primary difference between the Chai class (first definition) and the Chai class (second definition with pointers and destructor)?
7. Identify and explain the purpose of virtual functions and override keywords in the Tea and greenTea class hierarchy.
8. How does the masalatea class demonstrate the final keyword in its brew method? What does final imply?
9. Describe the functionality of the BankAccount class. What are its main operations and how does it ensure data integrity?
10. Explain the difference in how the prepareOrder functions (the two distinct versions) handle and return data.

Answer Key

1. Explain the purpose of the relational() function. The relational() function is designed to award a "badge" based on the number of cups a user has bought. It uses conditional statements (if, else if, else) to check if numOfCups meets certain criteria (greater than 20 for Gold, between 10 and 20 for Silver).
2. Describe how a simple ascending star pattern is generated using nested loops, as seen in multiple examples. A simple ascending star pattern is generated using an outer loop for rows and an inner loop for columns. The inner loop's iteration count depends on the outer loop's current iteration, causing more stars to be printed in each subsequent row.
3. What is function overloading, and where is an example of it in the provided code related to the sereveChai function? Function overloading allows multiple functions to have the same name as long as their parameter lists (number, type, or order of parameters) are different. In the code, sereveChai(int chai) and sereveChai(string tea) are examples of function overloading, distinguished by their parameter types.
4. Explain the concept of a "copy constructor" in C++ with reference to the Chai class example. A copy constructor is a special constructor that creates a new object as a copy of an existing object. In the Chai class, the copy constructor Chai(Chai &other;) deep copies teaName by allocating new memory for the string and copying its content, preventing issues with shallow copies of dynamically allocated members.
5. How is dynamic memory allocation demonstrated in the prepareOrder function (the version using int* orders)? What is its importance? Dynamic memory allocation is shown with int *orders = new int[cups];. This allocates an array of integers on the heap, whose size is determined at runtime by the cups parameter. This is important for creating data structures whose size is not known until the program executes.
6. What is the primary difference between the Chai class (first definition) and the Chai class (second definition with pointers and destructor)? The primary difference lies in how they handle the teaName member. The first Chai class uses a string object directly, while the second Chai class uses a string* (a pointer to a string) which requires dynamic memory allocation in the constructor and a destructor to release that memory. The second also includes a copy constructor for proper deep copying.
7. Identify and explain the purpose of virtual functions and override keywords in the Tea and greenTea class hierarchy. Virtual functions enable polymorphism, allowing the correct version of a method to be called at runtime based on the actual object type, even when accessed via a base class pointer or reference. The override keyword explicitly indicates that a member function is overriding a virtual function in a base class, helping to catch errors at compile time.
8. How does the masalatea class demonstrate the final keyword in its brew method? What does final imply? The masalatea class uses final in void brew() const override final. The final keyword prevents any further overriding of this specific virtual function in derived classes. This ensures that the implementation of brew in masalatea is the definitive one for any class inheriting from masalatea.
9. Describe the functionality of the BankAccount class. What are its main operations and how does it ensure data integrity? The BankAccount class simulates a basic bank account, managing an accountNumber and balance. Its main operations are deposit() and withdrawl(). It ensures data integrity by checking that deposit amounts are positive and that sufficient balance exists for withdrawals, preventing invalid transactions.

10. Explain the difference in how the prepareOrder functions (the two distinct versions) handle and return data. One prepareOrder function (int *prepareOrder(int cups)) dynamically allocates an array on the heap and returns a pointer to it, allowing it to return multiple calculated values. The other prepareOrder function (int prepareOrder(int cups)) calculates a value within a loop but only returns the last calculated orders value as a single integer, effectively losing all previous computations.

Essay Format Questions (No Answers)

1. Analyze the various C++ class implementations provided (e.g., Chai, Form, BankAccount, Tea, Movie, Book, Employee, Phone). Discuss the common object-oriented programming principles demonstrated across these examples, such as encapsulation, constructors, and methods. What are the advantages of using classes in these scenarios?
2. The source code includes several examples of pattern printing using nested loops. Choose two distinct patterns (e.g., a simple star pyramid and the more complex diamond-like pattern) and provide a detailed explanation of the logic behind their implementation, specifically focusing on how the outer and inner loops control the output and handle spaces and characters.
3. Compare and contrast static arrays (like chaiServed and tea in the totalChaiServed example) with dynamic memory allocation using pointers (like in the Chai class with teaName or prepareOrder returning int*). Discuss the advantages and disadvantages of each approach in different programming contexts.
4. Discuss the importance of constructors, copy constructors, and destructors in C++ classes, using the Chai class (the one with dynamic memory allocation) as a primary example. Explain how these special member functions contribute to proper resource management and prevent common programming errors like memory leaks or dangling pointers.
5. Polymorphism is demonstrated through the Tea class hierarchy using virtual functions. Explain how polymorphism is achieved in this context and why it is a powerful feature in object-oriented design. Provide examples from the greenTea and masalatea classes to illustrate your points.

Glossary of Key Terms

- #include: A preprocessor directive used to include header files, which provide declarations for functions and classes.
- using namespace std;: A directive that makes names from the std (standard) namespace accessible without the std:: prefix.
- int main(): The entry point function for any C++ program. Execution begins here.
- cout: An object of the ostream class used for printing output to the console.
- cin: An object of the istream class used for reading input from the console.
- endl: A manipulator used with cout to insert a newline character and flush the output buffer.
- int: A fundamental data type used to store integer (whole number) values.
- string: A data type from the std namespace representing sequences of characters (text).
- double: A fundamental data type used to store double-precision floating-point numbers.
- float: A fundamental data type used to store single-precision floating-point numbers.
- char: A fundamental data type used to store single characters.
- if-else if-else: Conditional statements used to execute different blocks of code based on whether specified conditions are true or false.
- for loop: A control flow statement that allows code to be executed repeatedly a specific number of times.
- while loop: A control flow statement that repeatedly executes a block of code as long as a given condition is true.
- class: A blueprint for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions).
- object: An instance of a class.
- public: An access specifier that makes class members accessible from outside the class.
- private: An access specifier that makes class members accessible only from within the class itself.
- protected: An access specifier that makes class members accessible within the class and by derived classes.
- constructor: A special member function that is automatically called when an object of a class is created, used for initializing the object's state.
- destructor: A special member function that is automatically called when an object is destroyed, used for releasing resources acquired by the object.
- copy constructor: A constructor that creates a new object by copying an existing object of the same class.
- vector: A dynamic array from the Standard Template Library (STL) that can grow or shrink in size.
- function overloading: The ability to define multiple functions with the same name but different parameter lists.

- pointer: A variable that stores the memory address of another variable.
- new operator: Used for dynamic memory allocation on the heap.
- delete operator: Used to deallocate memory previously allocated with new.
- this keyword: A pointer to the current object within a member function.
- virtual function: A member function declared in a base class that can be redefined (overridden) by derived classes, enabling polymorphism.
- override keyword: Explicitly indicates that a member function is intended to override a virtual function in a base class.
- final keyword: Used to prevent a virtual function from being overridden in any further derived classes, or to prevent a class from being inherited.
- static_cast: An operator used for explicit type conversion during compilation.
- inheritance: A mechanism where one class (derived class) acquires the properties and behaviors of another class (base class).
- polymorphism: The ability of objects of different classes to be treated as objects of a common base class, often achieved through virtual functions.
- encapsulation: The bundling of data (attributes) and methods (functions) that operate on the data into a single unit (class), and restricting direct access to some of the object's components.
- array: A collection of elements of the same data type, stored in contiguous memory locations and accessed by an index.