**四川大学期末考试试题（闭卷）**

**（2022——2023学年第 2 学期） A卷**

课程号：304024030 课序号：08 课程名称：高级语言程序设计-Ⅱ 任课教师：赵启军 成绩：

适用专业年级：计算机学院2023级 学生人数：12 印题份数： 学号： 姓名：

|  |
| --- |
| **考 生 承 诺**  我已认真阅读并知晓《四川大学考场规则》和《四川大学本科学生考试违纪作弊处分规定（修订）》，郑重承诺：  1、已按要求将考试禁止携带的文具用品或与考试有关的物品放置在指定地点；  2、不带手机进入考场；  3、考试期间遵守以上两项规定，若有违规行为，同意按照有关条款接受处理。  **考生签名：** |
| **Ⅰ. Multiple Choice (3 points each, 30 points in total)**  1. What are the values of the following expressions? 5 / 2, 5 == 5u, ( 5 >> 3 ) **（ B ）**  A. 2,true,1  B. 2,true,0  C. 1,false,0  D. 1,true,1  2. Which keyword can define a function or a variable in a class that can be shared between all objects of this class?**（ B ）**  A. const  B. static  C. template  D. virtual  3. What type of variable can create an alias of another variable? ( A )  A. Reference variable  B. Pointer variable  C. Auto variable  D. Const variable |

|  |
| --- |
| 4. std::cout is**（ D ）**  A. a type or class  B. a function  C. a macro  D. an instance of a class  5. Which of the following allows a type to be a parameter of a method, class, or interface in C++? **（ A ）**  A. Template  B. Hierarchy  C. Polymorphism  D. Overloading  6. What will the program do when we use ‘new‘ to allocate a memory for a class? **（ A ）**  A. Allocate the memory and call the constructor.  B. Allocate the memory and call the destructor.  C. Allocate the memory and call the non-special member function.  D. Allocate the memory and do nothing after that.  7. The iterators for std::list,std::vector are: **( B )**  A. bidirectional, bidrectional  B. bidirectional, random-access  C. random-access, bidirectional  D. random-access, random-access  8. The expression std::vector<int> vec( 3 ) creates a container that contains three 0s. What  type of constructor does this expression call?**（ C ）**  A. Copy Constructor or Copy Assignment Operator  B. Default Constructor  C. Non-Default Constructor (Parameterized Constructor)  D. Move Constructor  9. What is the output of the following code or is there something wrong?**（ A ）**    A. 10  B. 20  C. The output is undefined.  D. Compilation error.  10. What is the output of the following code or is there something wrong? (Don’t care about head files)**（ C ）**    A. 1  B. 0  C. The output is undefined.  D. Compilation error. |

|  |
| --- |
| **Ⅱ. Fill in the Blanks (4 points each, 20 points in total)**  1. nullptr has type \_\_ nullptr\_t \_\_\_.  2. \_\_ Lambda \_\_\_ expressions can capture variables in their scope and are anonymous function objects. We can understand them as anonymous inline functions that can be used to replace standalone functions or function objects, making the code more readable.  3. In C++, there is a container that preserves a collection of unique elements, where the position of the elements is determined by the hash of their values. This container is known as \_\_\_ unordered\_set \_\_.  4. For each virtual class, a \_\_ virtual table \_\_\_ is defined as a lookup table of functions to resolve function calls and support dynamic dispatch.  5. The \_\_ auto \_\_\_\_ keyword specifies that the type of the variable will be automatically deduced by the compiler (from its initializer).  **Ⅲ. Concept Explanation (5 points each, 20 points in total)**   1. What are exceptions? How to throw and deal with exceptions and output the exception type in C++?   Exceptions are errors between run-time. Using keyword throw to throw an exception, and using keywords try and catch to deal the exceptions.   1. What is RTTI(Run-Time Type Information)? Try to explain this with the example of virtual methods in C++.   RTTI is a mechanism that allows the type of an object to be determined at runtime.  Class A has a virtual method func(), and class B, derived from class A, overrides this method. If we have a reference variable of type A referring to an instance of B and call func(), the func() in class B will be called.   1. Explain the main differences between shared\_ptr, unique\_ptr, weak\_ptr.   The std::shared\_ptr permits users to create multiple smart pointers that all point to the same object.  The std::unique\_ptr restricts users to creating only a single smart pointer for each unique object.  The std::weak\_ptr, which is typically used in conjunction with std::shared\_ptr, merely serves as a window to a std::shared\_ptr object and cannot access or modify the object.   1. List all the usages of keyword friend and give them brief introductions   A class A is a friend of another class B, with declaring friend class B in class A, means class B can use all public or non-public members of class A.  A class A is a friend of a function func(),with declaring friend return-type func(…) in class A, with, means that in function func(), we can use all public or non-public members of class A. |

|  |
| --- |
| **Ⅳ.** **Short Answer Questions (6 points each, 30 points in total)**   1. Give the result of following program with inputs below.     Output:  Is  there  a  template  function  in  the  program?   1. A copy constructor creates a new object as a deep copy of an exsisting object.      1. Copying plain data is a deep copy, with the value of a variable copied. Copying pointers just copy the address of one pointer, the data one pointer points does not be copied. 2. First: a class contains a pointer. Second: a class contains a reference. 3. It is not a copy constructor. Because during the process of parameter passing, it has a copy. If it is a copy constructor, it has a loop. 4. Read and complete the following program. There are 6 blanks in the program marked Q1 to Q6.         Q1: nullptr  Q2: T{} or T()  Q3: value  Q4: const my\_vector<T>&  Q5: delete[] m\_data  Q6: expension()   1. Can member functions of a class access static variables, and can static member functions of a class access ordinary variables? Why?   A member function can access static variables because static variables are not bound to class instances, it is shared by all objects of the class.  A static member function can’t access ordinary variables because ordinary variables belongs a special instance, however static functions are not bound to class instances.   1. There are three template classes matrix, invertible\_matrix, orthogonal\_matrix.        1. A reversible matrix (invertible matrix) has all the properties of a general matrix, and an orthogonal matrix always has an inverse matrix (its transpose matrix is the inverse matrix), so an invertible matrix can be derived from a general matrix, and an orthogonal matrix can be derived from an invertible matrix. 2. All matrices should support addition and multiplication operations, so these two operations should be defined in the matrix class. These operations do not change for the three types of matrices, so there is no need to override them. Only invertible matrices can perform inversion, so it should be defined in the invertible matrix class, and since the inversion of an orthogonal matrix can be achieved through transposition, it needs to be overridden. For a general matrix, division by an invertible matrix should be defined, so each type of matrix should have a division operation, and the division operation only needs to call the inv function of the corresponding parameter matrix. So there is no need to rewrite from the base class. |
|  |