**Roll No…………….. Total No. of Pages:……**

**ST-3 (SET-III)**

**4th SEMESTER 2022-23**

**22CS006- Object Oriented Programming**

**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. In C++, can a virtual function be static?
   1. Yes
   2. **No**
2. Which type of inheritance is required to implement virtual base classes in C++?
   1. Single inheritance
   2. **Multiple inheritance**
   3. Hierarchical inheritance
   4. Multilevel inheritance
3. How do you define a class template in C++?
   1. Using the keyword "class" followed by the template parameter list
   2. Using the keyword "template" followed by the class definition
   3. Using the keyword "generic" followed by the class name
   4. **Using the keyword "template" followed by the class name**
4. To catch multiple types of exceptions in C++, we can use:
   1. multiple try blocks
   2. if-else statements
   3. **nested catch blocks**
   4. switch statements
5. Which block should be used to handle exceptions in C++?
   1. **catch**
   2. try
   3. throw
   4. except
6. Can a class have both a virtual and a pure virtual destructor?
   1. **Yes, it is common practice**
   2. No, it is not allowed
   3. Only if the class is marked as final
   4. Only if the class is marked as abstract
7. Which type of polymorphism is achieved through function overloading in C++?
   1. **Compile-time polymorphism**
   2. Runtime polymorphism
   3. Static polymorphism
   4. Dynamic polymorphism
8. If a class inherits an abstract class and does not provide definitions for all pure virtual functions, it becomes:
   1. Concrete class
   2. **Abstract class**
   3. Static class
   4. Interface class
9. Which type of binding is determined by the type of the object pointed to or referenced, not by the type of the pointer or reference?
   1. Early binding
   2. Late binding
   3. **Virtual binding**
   4. Static binding
10. How is a template function overloaded in C++?
    1. **By providing multiple template parameter lists**
    2. By providing different return types for the same template function
    3. By providing different template parameter names
    4. By providing different function names

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

1. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual void show() {

std::cout << "Base ";

}

};

class Derived : public Base {

public:

void show() override {

std::cout << "Derived ";

}

};

int main() {

Base\* b = new Derived;

Derived\* d = new Derived;

b->show();

d->show();

return 0;

}

* 1. Base Base
  2. **Derived Derived**
  3. Base Derived
  4. Derived Base

1. What is the output of the following code?

#include <iostream>

class Shape {

public:

virtual void draw() const {

std::cout << "Drawing shape" << std::endl;

}

};

class Circle : public Shape {

public:

void draw() const override {

std::cout << "Drawing circle" << std::endl;

}

};

int main() {

Circle c;

const Shape\* s\_ptr = &c;

s\_ptr->draw();

return 0;

}

* 1. Drawing shape
  2. **Drawing circle**
  3. Compiler error
  4. Runtime error

1. What happens when this C++ program is compiled?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

class A

{

int a;

public:

A(){}

};

class B: public A

{

int b;

public:

B(){}

};

void func()

{

B b;

throw b;

}

int main()

{

try{

func();

}

catch(B \*b){

cout<<"Caught B Class\n";

}

catch(A a){

cout<<"Caught A Class\n";

}

}

* 1. Caught B Class
  2. **Caught A Class**
  3. Compile-time error
  4. Run-time error

1. Which of the following is true about the following program

#include <iostream>

using namespace std;

template <class P, class Q, class R>

class A {

P x;

Q y;

R z;

static int count;

};

int main()

{

A<int, int, int> m;

A<char, char, char> n;

cout << sizeof(m) << endl;

cout << sizeof(n) << endl;

return 0;

}

* 1. Compiler Error: template parameters cannot have default values
  2. 12 6
  3. **12 3**
  4. 6 3

1. What will be the output of this program?

#include <iostream>

using namespace std;

template <int i>

void fun()

{

i = 20;

cout << i;

}

int main()

{

fun<10>();

return 0;

}

* 1. **Compile Error**
  2. 10
  3. 20
  4. 15

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) Create a game development framework with a virtual base class GameObject and derived class Character.

Implement a virtual function update() in the GameObject class, and override it in the Character class to update the

character's position and other attributes.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | GameObject\* obj1 = new GameObject(5, 5);  obj1->update(); | Character\* char1 = new Character(10, 10, 100);  char1->update(); | Character\* char1 = new Character(50, 50, 120);  char1->update(); |
| **Output** | Updating GameObject: (5, 5) | Updating Character: (11, 11), Health: 110 | Updating Character: (51, 51), Health: 130 |

Solution :

**#include <iostream>**

**using namespace std;**

**// Base class representing a GameObject**

**class GameObject {**

**protected:**

**int posX;**

**int posY;**

**public:**

**GameObject(int x, int y) : posX(x), posY(y) {}**

**// Virtual function to update the GameObject's position**

**virtual void update() {**

**// Base implementation, just print the position**

**cout << "Updating GameObject: (" << posX << ", " << posY << ")" << endl;**

**}**

**};**

**// Derived class representing a Character, inheriting from GameObject**

**class Character : public GameObject {**

**private:**

**int health;**

**public:**

**Character(int x, int y, int \_health) : GameObject(x, y), health(\_health) {}**

**// Override the update function to update the Character's position and health**

**void update() override {**

**// Simulate movement, increase health, or other character-specific updates**

**posX += 1;**

**posY += 1;**

**health += 10;**

**cout << "Updating Character: (" << posX << ", " << posY << "), Health: " << health << endl;**

**}**

**};**

**int main() {**

**GameObject\* obj1 = new GameObject(5, 5);**

**Character\* char1 = new Character(10, 10, 100);**

**// Calling the update function on both GameObject and Character objects**

**cout << "Before Update:\n";**

**obj1->update(); // Calls the base class implementation**

**char1->update(); // Calls the overridden function in Character**

**// Clean up the allocated memory**

**delete obj1;**

**delete char1;**

**return 0;**

**}**

Q17) Write a C++ program that performs mathematical operations and handles the logic\_error exception for invalid calculations.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | Enter two numbers: 5 0 | Enter two numbers: 7 3 | Enter two numbers: 0 0 |
| **Output** | Exception: Invalid input for division or logarithm! | Division Result: 2  Logarithm Result: 1.09861 | Exception: Invalid input for division or logarithm! |

Solution :

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main() {**

**int num1, num2;**

**double result;**

**cout << "Enter two numbers: ";**

**cin >> num1 >> num2;**

**try {**

**if (num2 <= 0) {**

**throw logic\_error("Invalid input for division or logarithm!");**

**}**

**// Perform mathematical operations**

**result = num1 / num2;**

**cout << "Division Result: " << result << endl;**

**result = log(num2);**

**cout << "Logarithm Result: " << result << endl;**

**} catch (const logic\_error& e) {**

**cout << "Exception: " << e.what() << endl;**

**}**

**return 0;**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) You are a skilled audio engineer working on an exciting project to enhance the sound quality of audio recordings. As part of this critical endeavor, you have been assigned the task of creating a function template to perform element-wise addition of two arrays representing audio signals captured from different microphones.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | {1, 2, 3}  {4, 5, 6} | {1.1, 2.2, 3.3}  {0.5, 0.6, 0.7} | {'a', 'b', 'c'}  {'.', '#', '@'} |
| **Output** | Result: 5 7 9 | Result: 1.6 2.8 4 | Result: Å à ú |

Solution :

**#include <iostream>**

**using namespace std;**

**// Function template 'arrayElementWiseAddition' performs element-wise addition of two arrays.**

**// The template takes three parameters: 'arr1', representing the first input array of type 'T',**

**// 'arr2', representing the second input array of type 'T', and 'result', the output array of type 'T' to store the results.**

**// The function iterates through the arrays and adds corresponding elements of 'arr1' and 'arr2' to 'result' array.**

**template <typename T, size\_t N>**

**void arrayElementWiseAddition(const T arr1[N], const T arr2[N], T result[N]) {**

**for (size\_t i = 0; i < N; i++) {**

**result[i] = arr1[i] + arr2[i];**

**}**

**}**

**int main() {**

**// Example 1: Perform element-wise addition of two integer arrays.**

**int intArr1[] = {1, 2, 3};**

**int intArr2[] = {4, 5, 6};**

**int intResult[3];**

**arrayElementWiseAddition<int, 3>(intArr1, intArr2, intResult);**

**cout << "Result: ";**

**for (int i = 0; i < 3; i++) {**

**cout << intResult[i] << " "; // Output: Result: 5 7 9**

**}**

**cout << endl;**

**// Example 2: Perform element-wise addition of two double arrays.**

**char charArr1[] = {'a', 'b', 'c'};**

**char charArr2[] = {'.', '#', '@'};**

**char charResult[3];**

**arrayElementWiseAddition<char, 3>(charArr1, charArr2, charResult);**

**cout << "Result: ";**

**for (int i = 0; i < 3; i++) {**

**cout << charResult[i] << " "; // Result: Å à ú**

**}**

**cout << endl;**

**return 0;**

**}**