Euler Motors Internship Test DTU(Software)

komalmeena_co21a4_64@dtu.ac.in Switch account

Object Oriented Programming & Computer Architecture

2 points

```
Which of the following best defines the concept of 'Polymorphism' in C++?
```

- ○ The ability of a function or operator to behave in different ways depending on the types of its arguments.
- ○ The ability to combine data and functions in a single unit.
- ○ The ability to create new classes from existing classes.
- ○ The ability to hide details of implementation from the user.

2 points

Which of the following is an example of dynamic polymorphism?

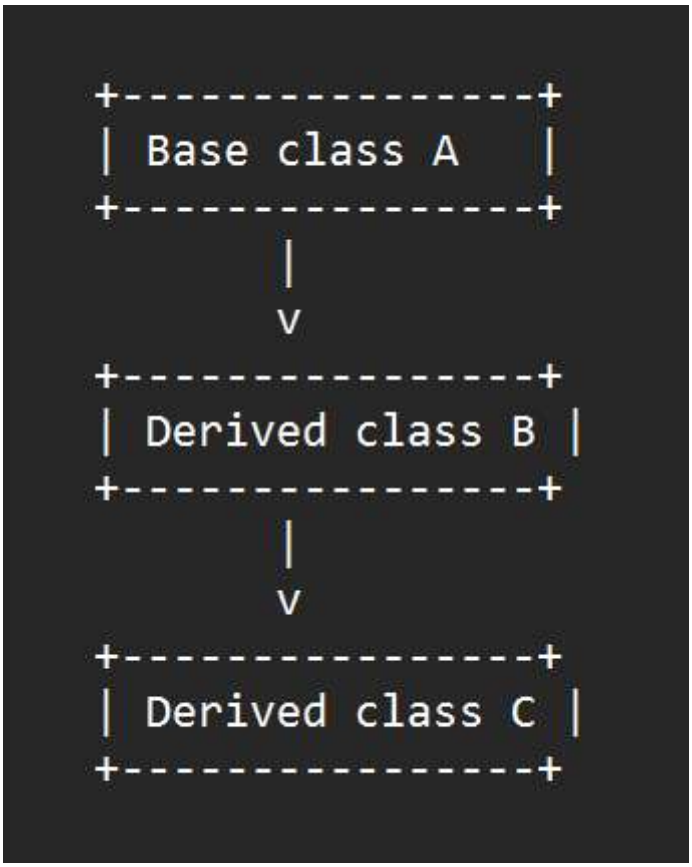
- Overloading a method with different parameter types in the same class.
- Defining a global function with the same name as another function in a different namespace.
- Accessing a static method using a class name instead of an object.
- Calling a method of a derived class through a base class reference at runtime.

2 points

What is the main advantage of inheritance in C++?

- It allows the derived class to reuse the code of the base class.
- It helps in hiding data members of the base class.
- It automatically generates constructors for the derived class.
- It prevents the creation of objects in the base class.

What type of inheritance is depicted in the following diagram? 2 points



- Single inheritance
- Multiple inheritance
- Multilevel inheritance
- Hierarchical inheritance

2 points

What is an abstract class in C++?

- ○ A class that cannot be instantiated
- ○ A class that must contain at least one pure virtual function
- ○ A class that can be instantiated
- ○ A class that contains only member functions

2 points

What is encapsulation in C++?

- ○ The ability to create new classes from existing classes
- ○ The ability to hide details of implementation from the user
- ○ The ability to perform different operations using the same function name
- ○ The ability to combine data and functions in a single unit

2 points

Which of the following is an example of compile-time polymorphism?

- Virtual functions
- Function overloading
- Inheritance
- Templates

2 points

Which of the following best describes a memory leak?

- When memory is reused from the heap without being properly released, leading to inefficient memory management.
- When memory is allocated dynamically but never deallocated, causing the program to consume more memory over time.
- When a program accesses memory that has already been freed, resulting in undefined behavior or crashes.
- When memory is freed multiple times, causing corruption or crashes in the program.

2 points

What is a dangling pointer?

- A pointer that points to a variable that has been deallocated
- A pointer that points to a variable that has not been allocated
- A pointer that points to a variable that is currently in use
- A pointer that points to a variable that has never been used

2 points

Which of the following describes the concept of "stack overflow"?

- When the stack pointer exceeds the stack bound
- When too many elements are pushed onto a stack
- When the program tries to use more memory than the stack has available
- All of the above

2 points

What is the primary purpose of a cache memory?

- To store frequently accessed data and instructions to speed up the operation of the computer
- To store all the data and instructions of a program
- To store user files and documents
- To manage input/output operations

2 points

What is a segmentation fault?

- An error caused by an infinite loop
- An error caused by dividing a number by zero
- An error caused by failing to declare a variable
- An error caused by accessing memory that the CPU cannot physically address

2 points

What is the function of the 'new' keyword in C++?

- To create a new variable
- To allocate memory on the heap
- To initialize a static variable
- To deallocate memory

2 points

Which of the following best describes the term "heap" in memory management?

- A region of memory used for static data
- A region of memory used for dynamic memory allocation
- A stack of function calls
- A type of data structure used in algorithms

2 points

```
Which operators cannot be overloaded in C++?
```

- It is possible to overload any operator in C++
- new, delete, ., .*, and !=
- ::, new, delete, sizeof, and ==
- ::, ., sizeof, .*, and ?:

Predict the output of the following code snippet 2 points

```
#include <iostream>
using namespace std;

class SyntaxWizard {
    public:
        SyntaxWizard()
        {
            cout<<"Constructor called ";
        }
        ~SyntaxWizard()
        {
            cout<<"Destructor called";
        }
};

int main(){
    SyntaxWizard *obj = new SyntaxWizard();
    return 0;
}
```

- ◯ Constructor called Destructor called
- ◯ Constructor called
- ◯ Destructor called
- ◯ error: conversion from ‘SyntaxWizard*’ to non-scalar type ‘SyntaxWizard’ requested

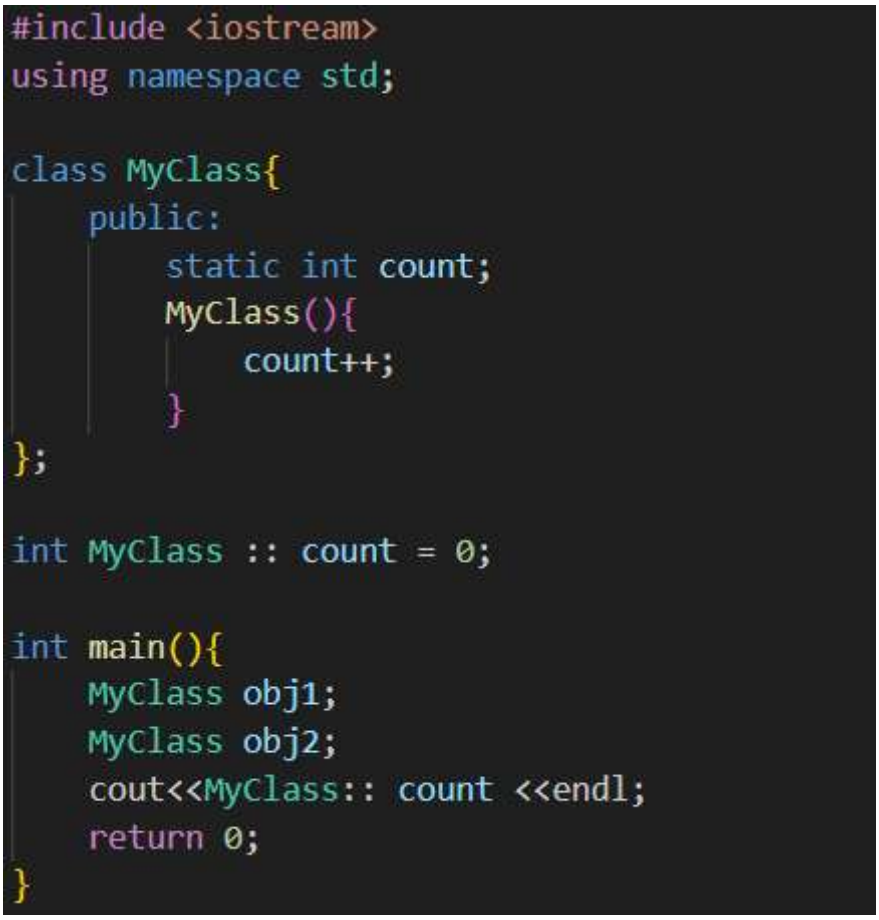
Predict the output of the following code snippet 2 points

```
#include <iostream>
using namespace std;

class SyntaxWizard {
    public:
        SyntaxWizard()
        {
            cout<<"Constructor called ";
        }
        ~SyntaxWizard()
        {
            cout<<"Destructor called";
        }
};

int main(){
    SyntaxWizard obj;
    return 0;
}
```

- ◯ Constructor called Destructor called
- ◯ Constructor called
- ◯ Destructor called
- ◯ error: conversion from ‘SyntaxWizard*’ to non-scalar type ‘SyntaxWizard’ requested

Predict the output of the following code snippet

4 points



```
#include <iostream>
using namespace std;

class MyClass{
    public:
        static int count;
        MyClass(){
            count++;
        }
};

int MyClass :: count = 0;

int main(){
    MyClass obj1;
    MyClass obj2;
    cout<<MyClass:: count <<endl;
    return 0;
}
```

- 1
- 2
- 0
- Compilation Error

Predict the output of the following code snippet 2 points

```
#include <iostream>
using namespace std;

class Base{
    public:
        virtual void show() = 0;
};

int main(){
    cout<<"Hello World! ";
    Base obj;
    cout<<obj.show();
    return 0;
}
```

- ○ Hello World!
- ○ Compilation Error: cannot declare variable ‘obj’ to be of abstract type ‘Base’
- ○ Hello World! 0
- ○ Compilation Error: pure virtual function ‘void Base::show()’ has no definition

Predict the output of the following code snippet

2 points

```
#include <iostream>
using namespace std;

class MyClass{
    public:
        int num = 5;

    public:
        MyClass(){
            this->num = 8;
        }

        MyClass(int num){
            this->num = num;
        }

        void display(int data){
            cout<<"num = "<<num;
        }
};

int main(){

    MyClass obj;
    obj.display(10);
    return 0;
}
```

- ○ num = 10
- ○ num = 8
- ○ num = 5
- ○ Compilation Error: invalid use of constructor

Predict the output of the following code snippet

2 points

```
#include <iostream>
using namespace std;

class MyClass{
    public:
        int num = 5;

    public:
        MyClass(){
            this->num = 8;
        }

        MyClass(int num){
            this->num = num;
        }

        void display(int num){
            cout<<"num = "<<num;
        }
};

int main(){

    MyClass obj;
    obj.display(10);
    return 0;
}
```

- ○ num = 10
- ○ num = 8
- ○ num = 5
- ○ Compilation Error: invalid use of constructor

Predict the output of the following code snippet 2 points

```cpp
#include <iostream>
using namespace std;

class Base{
    private:
        int x;
};

class Derived : public Base{
    public:
        void setX(int a){
            x = a;
        }
};

int main(){

    Derived d;
    d.setX(4);
    cout<<d.x;
    return 0;
}
```

- ○ 4
- ○ Compilation Error: ‘int Base::x’ is private within this context
- ○ No output
- ○ Compilation Error: invalid use of non-static data member

Predict the output of the following code snippet 2 points

```
class Base:
    def display(self):
        print("Base", end = " ")

class Derived(Base):
    def display(self):
        print("Derived", end = " ")

obj1 = Base()
obj2 = Derived()

obj1.display()
obj2.display()
Base.display(Derived)
```

- ○ Base Derived Derived
- ○ Base Derived Base
- ○ Base Base Derived
- ○ Base Derived AttributeError

Predict the output of the following code snippet 2 points

```
class Myclass:

    def __init__(self):
        self.quantity = 2

    def calc_price(self, quantity) -> float:
        return self.quantity * 10

obj = Myclass()
price = obj.calc_price(10)
print(price, type(price))
```

- ◯ 20 <class 'int'>
- ◯ 100 <class 'float'>
- ◯ 20.0 <class 'float'>
- ◯ 100 <class 'int'>

Back Next Clear form