CSE-1221 Mid Autumn 2022 Solution

Name: Ezaz Ahmed ID: C223009 Section: 2AM

1(a)

Ans:

Characteristics of OOP with examples:

Encapsulation: This refers to the ability of an object to hide its internal data from the outside world and expose only the necessary interfaces. This helps to prevent external interference and ensures that the object's internal state remains consistent.
Example: A bank account object can encapsulate its balance, and only provide interfaces for depositing and withdrawing funds.

Inheritance: This is the ability of one class to inherit the properties and behavior of another class. This allows for code reuse and enables the creation of new classes that are similar to existing ones, but with additional or modified functionality
Example: A Car class can inherit from a Vehicle class, and add specific properties and behaviors that are unique to cars.

Polymorphism: This is the ability of objects of different classes to be treated as if they were of the same class. This allows for more flexible and generic code that can handle a variety of different objects.
Example: A Shape class can have subclasses such as Circle, Square, and Triangle, and all of these objects can be treated as Shapes and manipulated in the same way.

Abstraction: This refers to the process of simplifying complex systems by breaking them down into smaller, more manageable pieces. This helps to make code more modular and easier to maintain
Example: An Employee class can abstract away the details of how an employee's salary is calculated, and provide a simple interface for getting and setting the salary.

1(b)

Ans:

The main differences between struct and class are:

1. **Struct are value types, while class are reference types.**
2. **Struct do not support inheritance, while class do.**
3. **Struct have default access modifiers of "public", while for class, it is "private".**
4. **Struct cannot have a default constructor, while class have a default constructor.**
5. **Struct are stack-allocated, while class are heap-allocated.**
6. **Struct have a fixed size, while class have a size that depends on their members.**
7. **Struct are copied by value, while class are copied by reference.**
8. **Struct are used for simple data types such as numbers, characters, and small collections of data, while class are used for more complex data types and behavior.**
9. **Struct can be faster than class for small data types because they do not require heap allocation and garbage collection.**
10. **Struct cannot be null, while class can be null.**

1(c)

Ans:

Problems associated with structured programming are solved by object oriented programming:

1. **Reusability: One of the primary benefits of OOP is the ability to reuse code. In structured programming, code is organized around procedures and functions. In OOP, code is organized around objects, which encapsulate both data and behavior. This allows developers to reuse objects in different contexts, which can save time and reduce the risk of errors.**

2. **Complexity management: OOP allows developers to manage complexity more effectively. With structured programming, as programs grow larger and more complex, the number of procedures and functions can become overwhelming, and it can be difficult to keep track of all the different pieces of code. In OOP, objects encapsulate data and behavior, which helps to simplify code and make it more manageable.**

3. **Data hiding: OOP provides the ability to hide implementation details from the rest of the program. This means that objects can be designed to have a public interface that is easy to use, while keeping the details of how the object works hidden from view. This makes it easier to change the implementation of an object without affecting the rest of the program.**

4. **Security:** In structured programming, it is difficult to protect data from accidental or intentional modification. In OOP, encapsulation provides an additional layer of security as the data can only be accessed or modified by the object's methods. This promotes security in the code.

1(d)
Ans:

The error in the given code is that the private member variable "f" of the student class is being accessed outside the class in the main function. As "f" variable isn't declared as public it cannot be accessed from outside the class.

1(d)or
Ans:

The error in the code is in the show(). Since show() is a member function of the student class, it should be defined inside the class.

2(a)
Ans:

The main goal of a friend function is to provide access to the private and protected members of a class to a non-member function or another class, without violating the encapsulation principle of object-oriented programming.

An appropriate example to illustrate the use of a friend function is written below:

```
#include <iostream>
 using namespace std;
 class A
{
 private: int l;
 public:
A(): l(2) { }
friend int C(A); //friend function
 };
 int C(A d)
{
d.l+=1;
return d.l;
 }
int main()
 {
 A d;
cout<<C(d)<<endl;
```

```
 return 0;
}
```

2(b)

Ans:

Two possible restrictions to inline functions are:

1. Function complexity: Functions that are too complex are not suitable for inlining because they can increase code size, memory usage, and execution time.

2. Recursive functions: Functions that call themselves recursively cannot be inlined because the function call stack is required to maintain the state of the function at each level of recursion.

Example:

```
#include<iostream>
using namespace std;
inline int A(int a)
{
   return a*a;
}
int main()
{
   cout<<A(2) <<endl;
   return 0;
}
```

2(c)

Ans:

(a) Output is : 10 10

(b) Output is : 4 and in some compiler it will show garbage value. As this-> is not used.

2(c) or

Ans:

The error in the code is due to an overloaded default constructor X().

3(a)
Ans:
Parameterized Constructor: **A constructor is called Parameterized Constructor when it accepts a specific number of parameters.**
A constructor function is invoked automatically when an object is created.

3(b)
Ans:
When two or more methods in a class have distinct parameters but the same method name, this is known as function overloading. One would overload a function to provide multiple ways to use the same function, with different parameter types or number of parameters, for better code organization, reusability, and ease of use.

A compiler identifies an overloaded function by looking at the number, type, and order of arguments in the function signature. If two or more functions have the same name but differ in the number, type, or order of their arguments, they are considered overloaded functions.
Example:

```
#include <iostream>
using namespace std;
void add(int a, int b)
{
  cout <<a + b<<endl;
}
void add(double a, double b)
{
   cout <<a + b;
}
 int main()
{
   add(1, 2);
   add(1.1, 2.2);
   return 0;
}
```

3(c)
Ans:
We use a copy constructor to create a new object from an existing object, with the same values, and is helpful in avoiding unwanted changes to the original object.

Example:

```
#include <iostream>
using namespace std;
class A
{
   int a;
public:
   A(int b)
   {
      a = b;
   }
   A(const A &o)
   {
      a = o.a;
   }
   void show()
   {
      cout <<a<< endl;
   }
};
int main()
{
   A o1(1);
   A o2 = o1;
   o1.show();
   o2.show();
   return 0;
}
```

3(c)or

Ans:

When a C++ class is created, the compiler automatically generates a default copy constructor, which performs a shallow copy of the object's data members. This means that when an object is copied using the default copy constructor, a bitwise copy of the object is created, with all data members copied over to the new object.

Example:

```
#include <iostream>
using namespace std;
class A
{
public:
   int x;
   double y;
   A(int a, double b)
   {
      x = a;
      y = b;
   }
};
int main()
{
   A obj1(1, 1.1);
   A obj2 = obj1; // default copy constructor called
   cout << "obj1.x = " << obj1.x << ", obj1.y = " << obj1.y << endl;
   cout << "obj2.x = " << obj2.x << ", obj2.y = " << obj2.y << endl;
   return 0;
}
```

In the given code, class A has two data members - an integer x and a double y. It also has a parameterized constructor which initializes x and y with the given values. In the main function, an object of class A (obj1) is created by calling the constructor with the arguments 1 and 1.1 .
Then, a new object (obj2) is created and initialized with obj1. Since no copy constructor is defined explicitly, the compiler provides a default copy constructor which performs a bitwise copy of obj1 to obj2. This means that each data member of obj1 is simply copied to the corresponding data member of obj2.
Finally, the values of x and y for both objects are printed using cout statements. The output shows that obj2 has been successfully initialized with the values of obj1, confirming that the default copy constructor has worked as expected.

Thanks Everyone Assalamualikum