Name: Arpit Singh

Subject : OOP

Student ID: 23112000014 05

Roll : 26

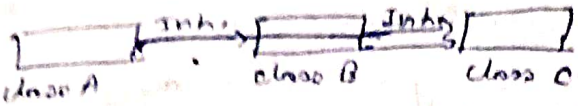
Sec - V

Dept. - B.tech (C.S.E)

Q. Define multiple inheritance with an example.

→ A kind of inheritance in which one class derives from a class which is already inherited some other class.

Diagrammatically;



[Here, 'C' shows multiple inheritance]

C derives both from both class A & B.

eg. class Animal {
public:
void get_data() { cout << "Animal breeds." << endl; }
};
class Mammal {
public:
void get() { cout << "Mammals breed." << endl; }.
};
class Cow : public Animal, public Mammal {
void get_info() { cout << "Cow is an animal as well as a mammal." << endl;
};

Q. Discuss the role of access specifiers in inheritance & show their visibility when they are inherited as public, private & protected.

→ Access specifier in inheritance control member visibility in derived classes:

a) public: Public member stay public in the derived class but the private remains private.

b) protected: protected stay protected in public or private inheritence but remains private in private ones.

c) private: private remain inaccessible in derived classes unless the specifier is changed.

Q. Explain different modes of inheritance.

→ • Single - A derived class inherits from a single base class.
• Multiple - A class inherits from more than one class.
• Multi-level - A class inherits from a class that has already inherited from some other class.
• Hierarchical - Multiple derived classes inherits from a single base class.
• Hybrid - A combination of different inheritances like single, multiple, multi-level, etc.

4. Describe Pure virtual func^n in details.

→ • Virtual func^n is a concept given to the base classes to create a func^n that can be overrided in any of the inherited class.
• virtual func^n works as any other class member by creating a interpass to carry out functionalities.
• virtual func^n pass the access just like any other member that is private & public of the same class.
• The base class also known as abstract class because of the presence of virtual func^n.

5. What is typecasting ?

→ Typecasting is a process to convert a ~~op~~ datatype to any other specific datatype. There are many ways to convert datatypes including implicit & explicit typecasting. Other measures are static typecasting but these measures are quite inappropriate for private data.

6. What are implicit & explicit type conversion ?

→ • Explicit type conversion –
• In this measure one datatype is converted to other using a cast operator.
• This is required b/w conversion of incompatible types or when a risk of data loss.
• Implicit type conversion –
• The compiler automatically converts one datatype to another when it is safe to do so.
• Usually happens when converting from a smaller dtype to a larger one.
• The conversion is done automatically without the need of explicit instructions.

7. What is try, catch & throw ? Explain with example.

→ Try, catch & throw are used for exception handling.
a) Try – Contains code that might throw an exception.
b) Catch – Used to raise an exception when an error occurs.
c) Throw – Catches & handles the exception thrown by the try block.

Example :

```
#include <iostream>
#include <stdexcept>
int divide (int a, int b) {
  if (b == 0) {
    throw
    std:: invalid_argument ("Division by 0 not allowed!");
  } return a/b;
}
int main () {
try {
int result = divide (10, 0);
}
catch (const std:: invalid_argument &e) {
   std:: cout << "Error : " << e. what () << std:: endl;
}
}
```

Develop a c++ program using func" template to find the products of two integers or floating type of data.

```
#include <iostream>
using namespace std;
template <typename T>
T multiply (T a, T b) {
   return a * b;
}
int main (){
int int1 = 5, int 2 = 10;
cout << multiply (int1, int 2) << endl;
float float1 = 5.5f, float2 = 3.2f;
cout << multiply (float1, float 2) << endl;
}
```

o/p:

50
17.6

What are the various types of files?

a) Source code files -
- These are files that contain the source code written in OOP lang., such as c++, Java, Python, C#, etc.

b) Header file -
- These contains declaration for classes, functions & other entities.

c) Object files -

- Object files are compiled version of source code files.

d) Executable files -

- These are created by linking object files. These files are runnable programs containing machine code.

10. What are the various ways in which a file can be opened? Explain by giving examples?

→
- ios::in - Open the file for reading.
  Eg - ifstream fil ("eg.txt", ios::in);
- ios::out - Open the file for writing.
  Eg - ofstream fil ("eg.txt", ios::out);
- ios::binary - Open the file in binary mode for reading & writing
  Eg - ofstream fil ("eg.txt", ~~ios::out~~ / ios::binary);
- ios::trunc - Truncates the file to 0 length when opened in ios::out.
  Eg - ofstream fil ("eg.txt", ios::out / ios::trunc);
- ios::ate - Opens the file & moves the pointer to the end for writing
  Eg - ofstream fil ("eg.txt", ios::ate);
- ios::in / ios::out - Open the file for both reading & writing
  Eg - ofstream fil ("eg.txt", ios::in / ios::out);

11. Write a C++ program to define a class Temp & write member func" to ask for temp. in fahrenheit & display in celsius.

→
```
#include <iostream>
using namespace std;
class Temp {
    float fahrenheit;
public:
    void get () {
        cout << "Enter temp. in °f : ";
        cin >> fahrenheit;
    }
    void display-c () {
        int c = (5/9) * (fahrenheit - 32);
        cout << "In °C : " << c << endl;
    }
};
```

```
int main () {
  Temp T;
  T.get();
  T.display-C();
  return 0;
}
```

12. Explain the concept of friend class with eg.

→ A friend class is class that is allowed with the access of the class that calls it. eg.

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

class T1 {
    int data;
public:
    void get() {
        cin >> data;
    }
    void set() {
        cout << data;
    }
    friend class T2;
};
class T2 {
public:
    T2 (int d) : data(d) { }
    void set2 () {
        cout << data;
    }
};
```

here, T2 can access the private 'data' attribute of 'T1' which is possible using the friend class.

13. What is polymorphism ?

→ One of the core principles of OOP, allows a func" or method to operate on different type of data or object.

Type of polymorphism:

a) Compile-time polymorphism: It is resolved during the compilation process. The most common types of this are function overloading or operator overloading.

b) Run-time polymorphism: It is resolved during execution. It typically involves inheritance & virtual func°.

14. Find errors if any:

→ a) cout <<" x = " x; [Error]

→ cout <<"x=" << x;

b) m = 5; n = 10; s = m + n; [Error]

~~→ m=5; n=10~~

→ int m = 5, n = 10, s = m + n;

c) cin >> x; >> y; [Error]

→ cin >> x >> y;

d) cout << \n "Name" << name; [Error]

→ cout << "\n Name" << name;

e) cout << " Enter value"; [No errors]

f) cin >> x; [No errors]